IBM DB2 11 for z/OS Performance Topics

Reduce processor time in CM and NFM

Improve scalability and availability

Evaluate impact of new functions

Paolo Bruni
Brian Baggett
Jeffrey Berger
Neena Cherian
Nguyen Dao
David Dossantos
Akiko Hoshikawa
Maggie Jin
Gopal Krishnan
Ping Liang
Cristian Molaro
Todd Munk
Mai Nguyen
Tomokazu Ohtsu
Bharat Verma
Lingyun Wang
Dan Weis
Chung Wu
David Zhang

Redbooks

ibm.com/redbooks

International Technical Support Organization

IBM DB2 11 for z/OS Performance Topics

July 2014

SG24-8222-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xxi.

First Edition (July 2014)

This edition applies to Version 11, Release 1 of DB2 for z/OS (program number 5615-DB2) and Version 11, Release 1 of DB2 Utilities Suite for z/OS (program number 5655-W87).

Contents

Figures

Examples

Tables

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at http://www.ibm.com/legal/copytrade.shtml

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
CICS®
Cognos®
DB2 Connect™
DB2®
Distributed Relational Database Architecture™
DRDA®
DS8000®
FICON®
FlashCopy®
HiperSockets™
HyperSwap®
IBM®
ILOG®
IMS™
InfoSphere®
iSeries®
Language Environment®
MVS™
OMEGAMON®
Optim™
pureXML®
RACF®
Redbooks®
Redpaper™
Redbooks (logo) ®
RETAIN®
RMF™
SPSS®
System z10®
System z®
Tivoli®
UC™
WebSphere®
z/OS®
z10™
zEnterprise®

The following terms are trademarks of other companies:

Netezza, NPS, and N logo are trademarks or registered trademarks of IBM International Group B.V., an IBM Company.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Preface

IBM® DB2® 10 for z/OS® provided a strong performance improvement that drove great value for IBM clients. DB2 11 for z/OS continues the trend by focusing on further CPU savings and performance improvement through innovations within DB2 and the synergy with IBM System z® hardware and software. Most of CPU reductions are built in directly to DB2, requiring no application changes. Enhancements often take effect after the REBIND in the case of static applications, while some enhancements require user actions such as Data Definition Language (DDL) updates.

DB2 11 CPU reduction can potentially provide significant total cost of ownership savings depending on the application mixture and type of transactions.

In this IBM Redbooks® publication we provide the details of the performance of DB2 11 for z/OS discussing the possible performance impacts of the most important functions when migrating from DB2 10. We include performance measurements made in the Silicon Valley Laboratory and provide considerations and high-level estimates. Keep in mind that results are likely to vary because the conditions and work will differ.

In this book, we assume that you are somewhat familiar with DB2 11 for z/OS. See *DB2 11 Technical Overview*, SG24-8180, for an introduction to the new functions.

Authors

This book was produced by a team of performance specialists working at the IBM Silicon Valley Laboratory, San Jose, California.

Paolo Bruni is a DB2 Information Management Project Leader at the International Technical Support Organization based in the Silicon Valley Lab. He has authored several IBM Redbooks publications about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During his years with IBM, in development and in the field, Paolo has worked mostly on database systems.

Brian Baggett is a senior software engineer with the IBM Silicon Valley Lab, where he works on the query performance team in DB2 for z/OS development. Brian is responsible for driving performance initiated development in each DB2 release working closely with DB2 developers. He has been working in the area of DB2 for z/OS performance since 1995, with emphasis on new query technology and solving customer performance problems.

Jeffrey Berger is a member of the DB2 for z/OS performance department at IBM Silicon Valley Laboratory. He has been with IBM for 34 years, mostly studying the performance of both the hardware and software of IBM mainframes, splitting his time between IBM storage, DB2, and the synergy between all of the above. Jeff has presented at numerous z/OS and DB2 conferences around the world and he has published several technical IBM Redpaper™ publications and articles with Computer Measurement Group, IDUG Solutions Journal, IBM Systems Magazine, and IBM Data Magazine. He has a Master's degree in Statistics, Operations Research, and Industrial Engineering from UC™ Berkeley.

Neena Cherian is a member of the DB2 for z/OS performance department at IBM Silicon Valley Laboratory. She has been with IBM for 24 years in all in the area of performance, out of which 5 years with DB2 performance. Her expertise in DB2 is in the area of OLTP performance analysis. Before joining DB2, she worked in Information Integration performance and Storage Subsystem performance. She has a Bachelor's degree in Computer Science from Temple University.

Nguyen Dao is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. He has been with IBM for 13 years, 4 years with DB2 for z/OS performance. He holds a Bachelor's degree in Computer Sciences from the University of Hawaii at Manoa (UH). His areas of expertise include DB2 for Linux, UNIX, and Windows performance, Information Server performance, Replication, and DB2 for z/OS distributed.

David Dossantos is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. He has been with IBM for 17 years, 8 years with DB2 for z/OS performance. He holds a Bachelor's degree in Computer Information Systems and Telecommunications from Cal State Hayward. He is the team lead for the DB2 Performance Regression team. His areas of expertise include DB2 XML performance, regression, and QA automation.

Akiko Hoshikawa is a Senior Technical Staff Member with the IBM Silicon Valley Lab, where she leads the performance engineers for DB2 for z/OS development. She is responsible for driving performance-initiated development in each DB2 release working closely with DB2 developers. She has been working in DB2 for the z/OS performance team since 1997, executing performance evaluations of DB2 new releases and supporting numerous customers' performance tuning and vendor benchmarks.

Maggie Jin is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. She has been with IBM for 11 years, 9 years with DB2 for z/OS query optimization and 2 years with DB2 for z/OS performance. Maggie has a Master's degree in Computer Engineering from University of California Irvine (UCI). Her areas of expertise include query optimization, query parallelism, and Query Accelerator.

Gopal Krishnan is a Senior Software Engineer at the IBM Silicon Valley, San Jose, California, US. He is the team leader for the DB2 for z/OS Performance Customer Support. He has expertise in the area of DB2 data sharing and transaction performance monitoring and tuning. He has presented in DB2 IOD, IDUG, SHARE, GUIDE, and IBM IMS™ technical conferences. He has a Master's degree in Computer Science from the State University of New York at Stony Brook and a Bachelor's degree in Electronics and Telecommunications Engineering from India.

Ping Liang is a member of the DB2 for z/OS performance department at the IBM China Development Laboratory. He has been with IBM for 10 years, 8 years with DB2 for z/OS Quality Assurance. He has a Bachelor's degree in Software Engineering from Huazhong University of Science and Technology. His areas of expertise include DB2 system performance analysis especially on insert performance tuning.

Cristian Molaro is an IBM Gold Consultant, an independent DB2 specialist, and a DB2 instructor based in Belgium. He has been recognized by IBM as an IBM Champion for Information Management in 2009 to 2013. His main activity is linked to DB2 for z/OS administration and performance. Cristian is co-author of several IBM Redbooks publications that are related to DB2. He holds a Chemical Engineering degree and a Master's degree in Management Sciences. Cristian was recognized by IBM as "TOP" EMEA Consultant at the IDUG EMEA DB2 Tech Conference Prague 2011.

Todd Munk is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. He has been with IBM for 19 years, all of them with DB2 for z/OS performance. He has a Bachelor's degree in Computer Science from the University of California San Diego (UCSD). His areas of expertise include distributed client connectivity and stored procedures.

Mai Nguyen is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. She has been with IBM for 30 years, the first 5 years with DB2 for z/OS Utility development and the rest are with DB2 for z/OS performance. She has a Bachelor's degree in Computer Engineering from San Jose State University. Her areas of expertise include DB2 Utility performance analysis and tuning.

Tomokazu Ohtsu is a Consulting IT Specialist in IBM Japan. He has 19 years of technical experience with IBM and has worked primarily with DB2 for z/OS. He has worked with many major customers in Japan designing and implementing DB2 for z/OS systems. He has experience working in the DB2 for z/OS performance department as an assignee at the IBM Silicon Valley Laboratory.

Bharat Verma is a Performance Analyst in the DB2 for z/OS performance department at IBM Silicon Valley Laboratory. He has more than 20 years of technical experience in Performance Analysis, Capacity Planning, Performance Tuning, Database Design, and Migration projects across multiple hardware platforms and operating systems. He has a Bachelor's degree in Electrical Engineering from Delhi College of Engineering, India.

Lingyun Wang is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. She has been with IBM for nine years, five years with DB2 for z/OS performance. She holds a Master's degree in Software Engineering with the University of Nebraska-Lincoln (UNL). Her areas of expertise include query optimization, IBM DB2 Analytics Accelerator, Replication, and XML.

Dan Weis is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. He has been with IBM for 45 years, more than 31 years in DB2. He has a BA in Math from Washington State University, 1968. His areas of expertise include DB2 problem analysis and tool development. Dan's contributions in this publication include the extended RBA and SQL statement storage.

Chung Wu is a member of the DB2 for z/OS performance department at the IBM Silicon Valley Laboratory. He has been with IBM for 24 years, 15 years with DB2 for z/OS performance. He has a Ph.D. degree in Mechanical Engineering from Stanford University. His areas of expertise include DB2 data sharing and online transaction processing (OLTP) performance analysis and tuning.

David Zhang is a DB2 for z/OS Performance Analyst in the IBM Silicon Valley Laboratory, US. He has 10 years of experience in DB2 for z/OS performance. He graduated from Brigham Young University with a Master's degree in Accountancy with information system emphasis. His areas of expertise include Master Data Management, IBM WebSphere® Portal, security, and distributed access to DB2.

Thanks to the following people for their contributions to this project:

Le Cai
Mengchu Cai
John Campbell
Steve Chen
Jason Cu
Cathy Drummond
Dan Kozin

Allan Lebovitz
Dave Levish
Leilei Li
Fenghui Li
Bob Lyle
Esther Mote
Jim Pickel
Emily Prakash
Terry Purcell
Haakon Roberts
Akira Shibamiya
Bart Steegmans
John Tobler
Steve Turnbaugh
Ping Wang
Julie Watts
Maryela Weihrauch
Chongchen Xu
Jay Yothers
IBM Silicon Valley Lab, San Jose

Brenda Beane
Seewah Chan
Elpida Tzortzatos
IBM Poughkeepsie

Jasmi Thekveli
IBM Cincinnati

Yong Dong Wu
Chen Xin Yu
IBM China

Richard Corrihons
Eric Romain
IBM Montpellier

Jasmi Thekveli
IBM Cincinnati

Andy Perkins
ZPABI IBM

Ying Chang
Frank Vitro
IBM - Retired

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our books to be as helpful as possible. Send us your comments about this book or other IBM Redbooks publications in one of the following ways:

- Use the online **Contact us** review Redbooks form found at:

  ibm.com/redbooks

- Send your comments in an email to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- Find us on Facebook:

  http://www.facebook.com/IBMRedbooks

- Follow us on Twitter:

  http://twitter.com/ibmredbooks

- Look for us on LinkedIn:

  http://www.linkedin.com/groups?home=&gid=2130806

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

  https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm

- Stay current on recent Redbooks publications with RSS Feeds:

  http://www.redbooks.ibm.com/rss.html

1

Introduction

Performance improvement is the primary focus in IBM DB2 11 for z/OS. Performance optimization in DB2 11 for z/OS influences a wide range of workloads, especially batch and queries. Performance improvements include potential CPU and cost reduction by simply migrating to DB2 11 conversion mode (CM) and REBIND, while new enhancements may require new-function mode (NFM) and user's action.

In this chapter, we provide an overview of the expectations for different types of workloads based on laboratory measurement and early usage experiences.

This chapter contains the following topics:

- Executive summary
- DB2 11 for z/OS at a glance
- Performance measurements for DB2 11 for z/OS
- Performance after migration
- How to read and use this book

1.1 Executive summary

DB2 10 for z/OS provided strong performance improvements, which drove great value to DB2 users. DB2 11 continues the trend by focusing on further CPU savings and performance improvement through innovations within DB2 and the synergy with System z hardware and software.

Most of the DB2 11 CPU reductions are built in directly into the DB2 engine and require no application changes. Enhancements often take effect after REBIND in the case of static applications or PREPARE for dynamic, while some enhancements require user actions such as DDL changes.

DB2 11 CPU reduction can potentially provide significant total cost of ownership savings depending on the application mixture and type of transactions.

In this IBM Redbooks publication, we provide the details of the performance of DB2 11 for z/OS discussing the possible performance impacts of the most important functions when migrating from DB2 10. We include performance measurements made in the Silicon Valley Laboratory and provide considerations and high-level estimates. Readers need to keep in mind that results are likely to vary because the conditions and workloads will differ.

1.1.1 Performance summary

A significant amount of optimization is done within DB2 11 to achieve CPU usage reductions, while providing business resiliency and industry-leading high availability features. Further scalability improvement is accomplished by taking advantage of the latest performance features in System z hardware and software. Consistent performance is important to DB2 users, and features are added to provide the relief of data administration work by reducing the frequency of running REORG.

CPU reductions

CPU reductions with DB2 11 can be seen in wide areas of SQL processing. In this publication, these DB2 CPU savings are analyzed by comparing them to the same workloads run on DB2 10.

DB2 online transactional workloads can expect CPU savings of up to 10% depending on the workload characteristics. Update, insert, and delete intensive batches run with less CPU cost due to a more efficient log processing starting in CM.

With improved decompression processing and optimizer enhancements, complex reporting queries can see up to 40% CPU savings. Queries executed in the DB2 Analytics Accelerator also see up to 10% CPU reduction in DB2 for z/OS.

REBIND is needed to obtain the best performance in the static applications. Additional CPU reduction can be achieved by taking advantage of new functions for areas such as log replication capture, IBM pureXML® or declared global temporary table (DGTT) usage.

The increase in System z Integrated Information Processor (zIIP) eligibility can contribute to cost reduction as most of asynchronous SRB processing under DB2 address spaces as well as additional utilities processing in RUNSTATS and LOADs have become zIIP eligible.

Scalability improvements

One of the most important scalability enhancements in DB2 11 is the ability to convert to a larger RBA/LRSN format. The extended RBA/LRSN provides up to 1 yottabyte of RBA record addressing capability or up 2**80 bytes of LSRN addressing. This provides not only continuous availability but performance improvement in the case of data sharing environment by completely eliminating the need of spinning to establish an LRSN unique value.

DB2 11 takes the advantage of the latest system z technology adopting larger page frame support and new Coupling Facility Control Code (CFCC) functions to prepare for rapid growth in data and business transactions.

By eliminating the overhead that is associated with a large number of partitions, DB2 11 achieved practically no difference in performance when accessing the same data from an increasing range of partitions up to the maximum value of 4096.

Consistent performance and productivity enhancements

Business resiliency is a key component of DB2 for z/OS and the ability to provide consistently good performance without the need of user intervention is the goal of DB2.

DB2 11 reduces the need of running REORG to keep the performance consistent by providing automatic cleanup of pseudo deleted index entries and avoiding the creation of overflow records.

By providing the break-in capability and automatic optimization, users are able to afford a more widespread use of the RELEASE (DEALLOCATE) bind option.

DB2 utilities provide more parallelism in DB2 11. This improves productivity and reduces administrative overhead without performance impact.

1.1.2 High-level performance expectations

Many workloads should see a CPU reduction when migrating to DB2 11, especially for update-intensive batches, queries against compressed tables, or queries that can take advantage of optimizer improvements.

Some of the improvements are not triggered until after REBIND as they require new package structures. While most of the improvements are available in CM without any application update or schema changes, new functions are offered to provide further performance improvement in NFM.

Figure 1-1 on page 4 shows the range of performance improvements that are expected in DB2 11 when compared to DB2 10. This figure shows averaged results by workload type as evaluated by multiple sets of measurements.

Compared to the same workload running on DB2 10, you might see the following improvements:

- From 0% to 10% CPU reduction for online transaction processing (OLTP).
- From 0% up to 10% for update intensive batch workload.
- Complex reporting queries can see up to 25% CPU savings and up to 40% CPU savings when the queries are running against compressed tables.
- Utilities show from 0% to 7% CPU reduction depending on the type of utility.

Since workloads have different characteristics, we can only show the expectation as the range shown in Figure 1-1. If the workload includes more use of DB2 11 strong performance features, a higher value of CPU reduction could be observed.

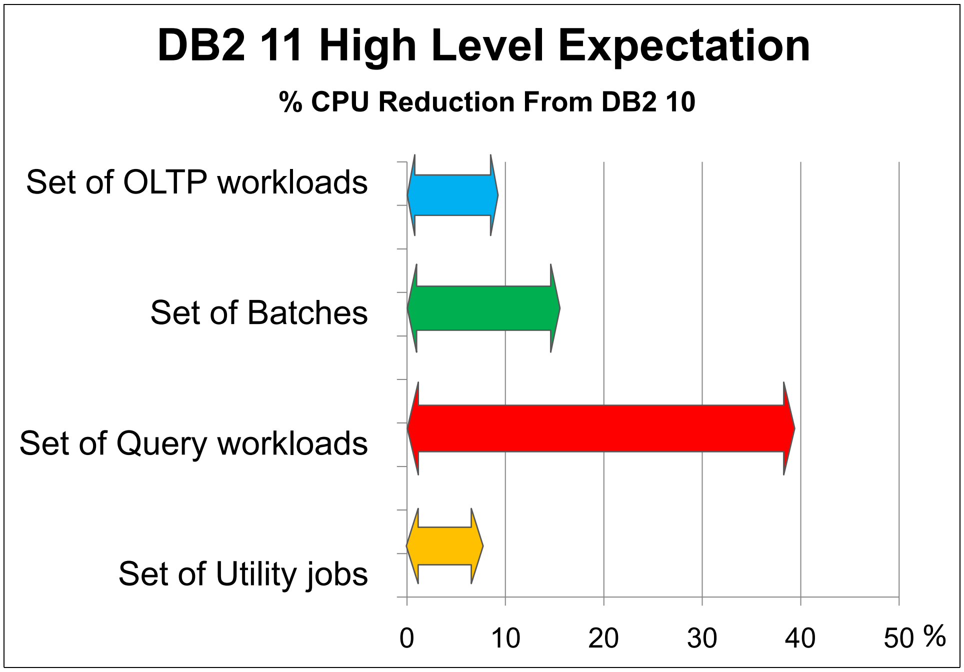

Figure 1-1 DB2 11 high-level performance expectations

Note: The DB2 11 performance expectations are based on comparing the execution of the same applications between DB2 10 NFM and DB2 11 CM or NFM, without changing any application nor DDL, but after REBIND.

1.2 DB2 11 for z/OS at a glance

This section provides the general introduction to key improvements delivered by DB2 11 for z/OS in these four areas:

- CPU savings and performance improvements
- Resiliency and continuous availability
- Enhanced business analytics
- Simper and faster DB2 version upgrades

Note: For more information about the new functions, refer to *DB2 11 Technical Overview*, SG24-8180.

1.2.1 CPU savings and performance improvements

DB2 11 delivers optimization in performance sensitive code paths. Most users can observe CPU reduction in OLTP, batch, and complex reporting queries without changing applications or schemas. Additional zIIP usage in DB2 address space processing can lead to operational cost reduction and can lead to overall performance improvement. The synergy with System z software and hardware contribute to further performance improvements in DB2 11.

Complex queries or queries against compressed tables can see significant improvement of up to 40% CPU reduction after REBIND.

In addition to transparent system optimization, new performance features are added to help application level performance.

In the area of data administration, DB2 11 provides autonomic features, which can reduce the need for REORG by providing the ability to keep the performance consistent.

1.2.2 Resiliency and continuous availability

DB2 11 drives new values in business resiliency through scalability improvements and fewer outages. This section summarizes these enhancements:

- Extended log RBA/LRSN format
- More online schema change and availability improvements
- Security improvements
- Work file usage control
- Data sharing improvements
- Utility enhancements
- Enhanced business analytics
- Simpler and faster DB2 version upgrades

Extended log RBA/LRSN format

Since DB2 version 1, the relative byte addressing (RBA) used as key value for log records has been 6 bytes. A 6-bytes field provides 256 TB of log record addressing. When the end of log RBA is reached, the user must take manual actions to reset the RBA back to zero.

In data sharing environments, the log record sequence number (LRSN), a 6-byte value derived from the TOD sysplex clock, is used for log records and this value runs out in the year 2042. The existing 6-bytes log format is called *basic log format*.

DB2 11 provides the support for 10 bytes of RBA and LRSN values and the new 10-bytes log format is called *extended log format*.

The conversion to the new format in case of migration consists of three logical steps to be run in NFM.

- Conversion of BSDS from 4 KB CI size to 8 KB by the job DSNTIJCB.
- Conversion of DB2 catalog and directory by the job DSNTIJCV.
- Conversion of objects (user and catalog table spaces and indexes) through REORG or other utilities.

After the conversion is completed, table spaces and indexes support up to 1 YB (yottabyte) in RBA or up to 2**80 in LRSN addressing.

DB2 11 starts up as a 10-bytes RBA/LRSN system but it functions perfectly without the full conversion to the extended RBA/LRSN. However, it requires some work in converting from 10-byte log records into 6 bytes when basic log format is still used. Although this overhead is not usually visible, there is some overhead using basic log format, especially with update intensive operations.

Log replication products such as IBM InfoSphere® Data Replication for DB2 for z/OS need to be able to read the 10 bytes log format from day one with DB2 11 and require having appropriate support already in CM.

In data sharing environments, the 6-byte LRSN can only provide a granularity of 16 microseconds. Starting with DB2 9, DB2 has reduced the needed level of uniqueness for log record sequence numbers, however, the sequential update or delete can still suffer the overhead (LRSN spin) necessary to obtain the unique log record sequence numbers on fast processors or multi-row inserts are used. The extended LRSN eliminates the need for spinning and therefore improves the sequential update/delete processing in a data sharing environment.

More online schema changes and availability improvements

DB2 11 improves online schema usability by lifting the restriction of point in time recovery to a point in time before the change materialization.

DB2 11 also lets you drop existing columns from a table via an ALTER TABLE ... DROP COLUMN RESTRICT SQL statement if there are no dependent objects defined on this column. The drop is materialized through REORG.

The need for planned outages is significantly reduced with online data repartitioning capabilities, including support for REORG REBALANCE SHRLEVEL, CHANGE, and online ALTER LIMITKEY.

SWITCH phase reduction and improved drain processing in online REORG deliver additional availability improvements.

The virtual storage constraint relief in DB2 10 made it possible for more packages to use the RELEASE (DEALLOCATE) BIND/REBIND option to reduce CPU cost of package allocation. However, executing packages bound with RELEASE (DEALLOCATE) on reusable threads blocks REBIND of the package, DDL, or online REORGs against the objects that the packages are accessing. Break in capability in DB2 11 provides the solution by allowing the execution of REBIND, DDL, or online REORG against persistent threads running packages with RELEASE (DEALLOCATE) by temporarily switching them to RELEASE(COMMIT).

Security improvements

DB2 for z/OS and System z continue to lead the industry with superior security and auditing capability. DB2 11 adds IBM RACF® exit enhancements, whereby external security managed by RACF administrators can now fully handle access to DB2 objects.

Furthermore, DB2 11 program authorization provides the ability to verify that an application program is correctly using a DB2 plan. DB2 11 removes restrictions related with aggregate functions improving the usability of the column masking functions introduced in DB2 10.

Work file usage control

In addition to the existing MAXTEMPS subsystem parameter, which determines the maximum amount of temporary storage in the work file database that a single agent can use, the agent level of work file usage monitoring is added in DB2 11.

The `WFSTGUSE_SYSTEM_THRESHOLD` subsystem parameter determines the percentage of available space in the work file database consumed by all agents before a warning message is issued.

Data sharing improvements

DB2 11 for z/OS provides a number of data sharing improvements to enhance continuous availability and to provide better performance.

Group buffer pool (GBP) write around protocol is a capability to bypass writing pages to the GBP in certain situations and write the pages directly to DASD instead, while still using the GBP to send buffer invalidate signals.

DB2 11 uses the protocol added in CFCC 17 level to reduce the situations where GBP fills up with changed pages, which can result in application slowdowns or, in severe cases, pages being written to the logical page list (LPL).

In addition, cast out processing becomes faster in DB2 11 by overlapping read and write.

The index tree modification, such as index split processing in data sharing, speeds up with DB2 11 by reducing the numbers of synchronous log force writes.

The RESTART LIGHT now can use CASTOUT option, which releases retained page set P-locks.

DB2 11 also improves the LPL recovery process by initiating automatic LPL recovery of objects at the end of both normal restart and restart light.

Utility enhancements

Partition level online REORG with NPI shows significant performance improvement in DB2 11 by sorting the NPI.

LOAD utility with a single input data set can now use parallelism without the need for splitting and sorting the input into multiple data sets. This change shows up to 80% elapsed time reduction.

Deep integration between DB2 and System z continues to bring value. Additional utilities processing in RUNSTATS, REORG, and LOAD now run under an enclave SRB and are therefore eligible for zIIP processing, driving greater efficiency and CPU savings. DB2 11 improves various utilities by adding or enhancing zIIP offloading capabilities.

Enhanced business analytics

Improved performance and DB2 CPU savings bring additional value in running business analytics alongside your core operational data. DB2 11 supports real time scoring directly within the transaction when used with the separately available IBM SPSS® offering.

Together with IBM DB2 Analytics Accelerator for z/OS and System z, you can efficiently co-locate OLTP and business analytics as one integrated workload with great cost effectiveness.

Temporal functions usability are further enhanced in DB2 11, and a transparent archive capability is added to increase productivity of application developers. Various SQL enhancements, including support for array data types in SQL PL and global variables, give more flexibility for application developers and simplify application porting.

There are a number of performance enhancements to XML processing in DB2 11, some of which are also retrofitted to DB2 10.

Simper and faster DB2 version upgrades

With the new application compatibility option to keep SQL functionality at a previous level, migration plans no longer must wait for the applications to be changed in order to deal with any incompatible SQL or XML changes before upgrading to the latest DB2.

Access path stability features are enhanced with DB2 11 by adding the APREUSE (WARN) option to reuse the access path whenever it is possible.

Simpler catalog migration helps your business get the value of DB2 11 faster.

1.3 Performance measurements for DB2 11 for z/OS

In this section, we discuss at high level the performance measurements and the strong performance points of DB2 11 for z/OS.

Figure 1-2 shows the measured CPU reduction when comparing DB2 11 to DB2 10 on sample benchmark applications.

Workloads are executed under the same strictly controlled environments. These workloads are commonly used for measuring release to release performance differences, and are not particularly built to take advantage of new functions provided by DB2 11. The values in this figure are established based on laboratory measurements and various measurements done by early users.

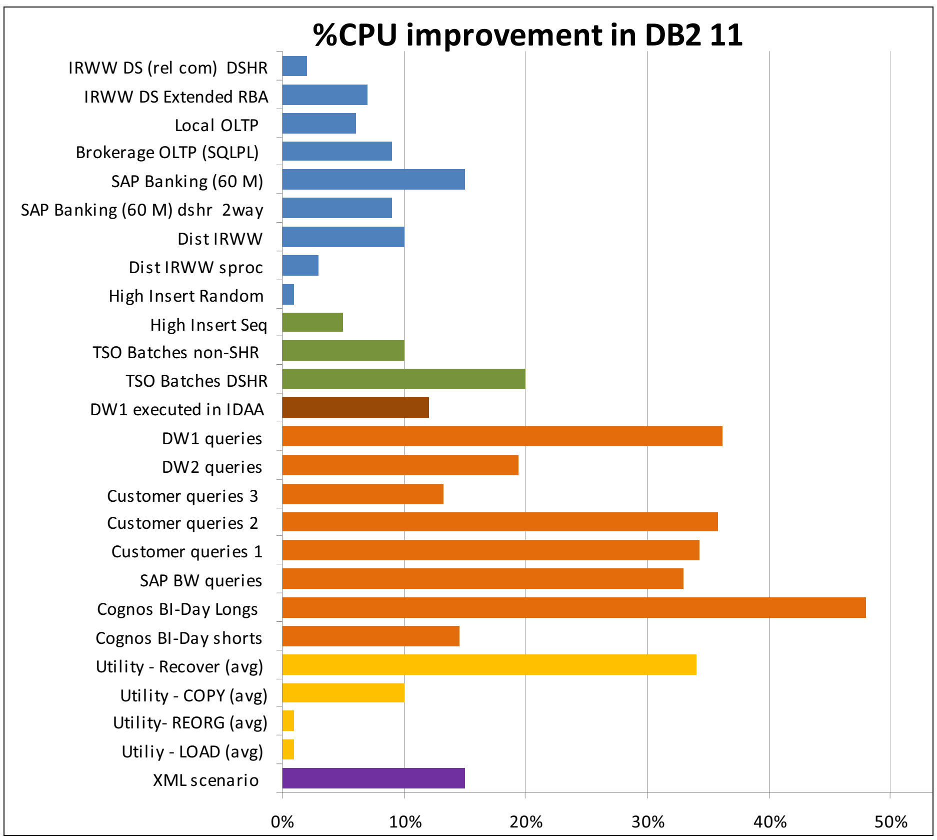

Figure 1-2 CPU reduction seen in DB2 11

The workloads that are mentioned in this figure can be summarized as follows:

- A set of IRWW workloads represents the simple, legacy type, OLTP workload that is typically executed through IMS or IBM CICS® as transaction servers.
- Distributed IRWWs are the same DB2 workloads that are executed through remote clients such as JDBC or remote stored procedure calls.
- Brokerage OLTP represents a complex OLTP with a mixture of lookup, update, and insert as well as reporting queries.

- SAP Banking Day Posting workloads are relatively simple SQL executions against an SAP banking schema.
- High insert workloads are concurrent inserts that are executed through more than 100 JCC T4 connections in data sharing.
- TSO batches are representing a batch workload executing various batch operations including SELECT, FETCH, UPDATE, INSERT, and DELETE in a loop.
- A set of two data warehousing queries workloads are executed serially mostly using CPU parallelism. Each query workload consists of a set of 20 to 100 queries per workload. Individual queries can see wide range of differences between DB2 10 and DB2 11, the average CPU reduction rate is shown here.
- IBM Cognos® BI-day short workload is a set of short running reporting queries concurrently executed.
- Utilities scenarios are set for common utility job executions.
- XML scenarios are also set of various XML workloads including OLTP, queries, batch, and utilities against XML data types.

Appendix B, "The IBM DB2 workloads" on page 297 shows some details about the workloads.

The next sections reference Figure 1-2 on page 8 and discuss:

- DB2 11 strong performance points
- Subsystem level performance
- Update performance
- Query performance
- Consistent performance and reduction of REORG frequency
- Application performance

1.3.1 DB2 11 strong performance points

As shown in Figure 1-2 on page 8, there is wide range of improvements observed in DB2 11 depending on the workload. Some workloads show equivalent performance when compared to DB2 10, while others show 10% or more improvement. This is because certain workloads are hitting more strong points (also called *sweet spots*) of DB2 11 while others do not.

A DB2 11 strong performance point is a new, or changed, DB2 11 feature that has the potential to provide significant performance improvements to an application or workload that is using it.

The strong points of DB2 11, that is the areas where we expect the higher level of performance improvement, include:

- Write intensive workloads
- Transactions with RELEASE(COMMIT) BIND option accessing one or a few partitions from tables with large number of partitions (>200 partitions) defined
- DDF transactions with large number of simple SQL statements per unit of work
- Transactions accessing large buffer pools
- FETCH or SELECT returning large number of rows and columns
- Queries against compressed tables
- Sort intensive queries
- Queries with access path improvement

- Queries accessing multiple partitions through DPSI
- Accelerator queries with large results set

1.3.2 Subsystem level performance

Online transaction processing is usually a large part of user workloads. Based on laboratory results, we have observed a range of CPU reduction up to 18%, but typical expectation is between 0% and 10% CPU savings for a workload running with RELEASE (COMMIT).

DB2 executable procedures (xProcs)

DB2 11 continues to add runtime optimizations called customized *procedures* or executable code sequences (or xProcs) for repeated operations, such as SELECT procedures, UPDATE procedures, and others. Some of these procedures are built during BIND/REBIND or PREPARE, and others are built at execution time.

These executable procedures are loaded into the DB2 DBM1 address space below the 2 GB bar, and are shared among threads in DB2 10. In DB2 11, they are now loaded above the 2 GB bar, as z/OS can support 64-bit executables. In addition, DB2 11 introduces two new customized procedures, one for column processing and one for sort processing.

The customized procedures for columns are built during BIND, REBIND, or PREPARE and can reduce the CPU usage on column processing during FETCH or SELECT operations. The larger the number of columns, the higher the CPU reduction becomes. Typically, applications select fewer than 20 columns and the reduction might not be visible for them. However, applications such as SAP, which selects some hundreds columns, can show a noticeable CPU reduction from these new column procedures. Static applications migrated from DB2 10 require REBIND to take advantage of this improvement.

The sort procedures are built during execution. Any sort operation can take advantage of them without REBIND. The larger the number of records sorted, the higher the CPU reduction becomes. In a query workload with 162 queries, we observed more than 7% CPU improvement due to the customized sort procedure usage for queries with large amounts of sort.

DB2 11 improves other several areas of sort operations. One of them is avoidance of the materialization of final merge to the sort work file. For a large sort operation, writing and reading the final results into the sort work file can impact the performance. Avoiding the materialization reduces the physical I/Os and getpages from work files.

System z synergy and cost reduction with zIIP

DB2 system tasks, such as prefetch reads and deferred writes, became zIIP eligible in DB2 10. DB2 11 adds more system tasks running under DBM1 or MSTR address spaces to be eligible for zIIP offload. They include asynchronous log read and write, and cast out processing. Customers should expect additional zIIP usage from pre-emptible SRBs in DBM1 and MSTR address spaces.

DB2 11 also expands zIIP eligibility in RUNSTATS, LOAD, REORG, and REBUILD INDEX processing to provide further utility CPU reduction.

DB2 11 takes further advantages of System z features such as Flash Express[1] and IBM zEnterprise® RoCE-Express[2] adapters:

---

[1] Flash Express is implemented with PCI Express (PCIe) adapters containing solid state drives configured in RAID 10 arrays across cable-connected pairs in I/O expansion drawers.

- With Flash Express installed and defined on zEC12 or zBC12, DB2 uses pageable 1 MB page frames for buffer pool control blocks to reduce the CPU usage of buffer access. With z/OS V2R1 and Flash Express, DB2 executable codes can be backed by 1 MB page frames.
- DB2 network LPAR to LPAR Type 4 communication through RoCE-Express improves network latency significantly compared to 10 Gbit Ethernet connections between CPs.

Scalability improvement

With DB2 10, there is a cost on accessing large number of partitions. The cost is not generally visible, unless an application access more than 1000 partitions and is using packages bound with RELEASE (COMMIT). DB2 11 addresses this cost and improves scalability. Transactions touching one or a few partitions from a partitioned table space with many partitions will show an improvement. This CPU reduction only applies to packages bound with RELEASE (COMMIT) as the packages bound with RELEASE (DEALLOCATE) do not incur this cost.

In general, the RELEASE (DEALLOCATE) bind option can result in CPU reduction by avoiding package allocation costs. However, the RELEASE (DEALLOCATE) option has not been used widely due to the virtual storage usage increase for holding the packages and thread information below the 2 GB bar. This became more feasible when DB2 10 moved most of thread storage and all of environmental descriptor manager (EDM) storages above the 2 GB bar.

However, there were still two usage concerns remaining for RELEASE (DEALLOCATE) in DB2 10:

- The inability to execute REBIND, DDL, or online REORG against the packages or objects that are touched by the packages running under thread reuse if the packages are bound with RELEASE (DEALLOCATE).

  DB2 11 introduces the ability of breaking into the persistent threads executing packages with RELEASE (DEALLOCATE) at COMMIT point.
- Potential CPU or storage impact from accumulation of objects-related information if the packages access large numbers of DB2 objects throughout the life of the thread.

  DB2 11 addresses this concern by introducing the detection of resource accumulations and their automatic release.

With these two concerns addressed, RELEASE (DEALLOCATE) BIND option can be used for more packages, local or remote, and consequentially reduce CPU utilization in DB2 11.

DDF enhancements

DDF transactions in DB2 11 can take advantage of a z/OS 1.13 Communications Server function called *Termination Notification for Asynchronous Socket processing*. This allows DB2 to use the optimized synchronous recv() socket processing but still be notified of any socket failure. With synchronous receive processing, DDF transactions can avoid the suspend/resume overhead and reduce both CPU and elapsed time when the transaction has more than a few SQL calls through TCP/IP communication.

The DB2 distributed IRWW benchmark workload showed 4% to 7% improvement using this feature. With z/OS 1.13 or later Communications Server, DB2 11 enables the feature automatically.

[2] The 10-Gigabit Ethernet (10GbE) Remote Direct Memory Access (RDMA) over Converged Ethernet Express feature (FC0411) is an RDMA capable network interface card exclusive to the zEC12 and zBC12 installed on the PCIe I/O drawer.

DB2 11 also provides a new continuous block fetch bind option, DRDACBF, to improve support for large result sets when a IBM DRDA® client on z/OS executes the queries. This is done by the DB2 requester opening additional sockets to the DB2 server for each read-only cursor.

1.3.3 Update performance

There are log processing enhancements in DB2 11 that can reduce the CPU usage during UPDATE, INSERT, DELETE, or MERGE operations. They are described in this section.

Output log buffers

The output log buffers are now allocated in high common area instead of the private region of the DB2 MSTR address space. This avoids the cross memory operations between DBM1 address space and MSTR address space during transaction log creation. The enhancement can provide significant CPU reduction in update intensive operations for data sharing and non-data sharing environments.

In addition to cross memory avoidance, log buffers can be optionally backed up by 1 MB page frames for further CPU reduction if there is enough 1 MB frames available during DB2 start.

Tested TSO batch jobs in a non-data sharing environment show about 10% CPU reduction compared to DB2 10 in basic log RBA format.

Impact of RBA/LRSN format

DB2 11 is built with a 10-bytes RBA/LRSN system. In CM or NFM with basic RBA/LRSN, DB2 converts 10 bytes RBA/LRSN to 6 bytes at externalization until the conversion is completed.

Although this overhead is not usually visible because it is balanced by the output log buffer enhancements, there could be further CPU reduction after the RBA/LRSN format is converted to 10 bytes in NFM by avoiding the conversion. This conversion requires both BSDS and the page set conversion.

In data sharing environments, further CPU reduction can be expected by eliminating the overhead from the LRSN spin effect. TSO batch jobs executed in a data sharing environment using extended LRSN shows up to 20% CPU reduction in sequential INSERT, UPDATE, or DELETE operations.

1.3.4 Query performance

Query workloads show a wider range of CPU reduction in DB2 11, 14% to 48%, due to numerous query CPU reduction items. In general, the more complex the queries are, the larger the rate of CPU reduction becomes, as there is more opportunities to trigger the improvements. This list shows the major query-related enhancements:

- Reduced CPU for processing compressed data pages
- Increased sparse index use and improved performance
- Index key skipping for DISTINCT, GROUP BY, and SET function
- Early out for MIN() or MAX(), inner table in join/sub queries
- Predicate rewrite and pushdown
- More use of in memory work files
- Avoidance of materializing final merge from top sort
- Reduced CPU for sequential access, especially for ACCESS_TYPE='R'
- DPSI access across multiple data partitions

Optimizer cost enhancements

Most enhancements require REBIND to trigger the improvement, but they can be triggered with REBIND using APREUSE (ERROR or WARN). Some items do require REBIND without using APREUSE to take advantages of optimizer enhancements.

Decompression enhancement

DB2 11 introduces a software algorithm to improve the decompression operation. This algorithm applies to compressed table spaces but not to indexes. The optimization decompresses only the necessary part of the record and this could result in a large amount of CPU reduction for queries returning a few columns but scanning large number of records. Laboratory and customer measurements show up to 70% CPU reduction for queries executing table space scan against compressed tables. Typically, since most of the predicates are evaluated or filtered through indexes, the range of CPU improvement would be 10-20%.

Sparse index enhancement

A sparse index is an array of join keys and RIDs. When no indexes are defined on a table and the table is not small, a sparse index can greatly improve performance.

DB2 continues to expand the usage of sparse index and in memory data cache since the sparse index had been introduced in DB2 V4. DB2 11 extends the usage of both sparse index and in memory data cache by removing some restrictions in DB2 10 and by optimizing the sparse index processing. The enhancement has large applicability. Testing shows that 10% to 20% of the queries now use sparse indexes across a broad range of query type workloads in DB2 11 after REBIND.

Optimizer cost model enhancement

Enhancements and adjustments to the DB2 11 optimizer result in more positive access path changes in DB2 11 when REBIND is done without using APREUSE.

1.3.5 Consistent performance and reduction of REORG frequency

Customers often observe performance degradation as the databases are continuously updated, inserted, or deleted over time. This is expected when the database becomes less clustered or index access gets more expensive as pseudo deleted entries increase as the rows are deleted. To keep the performance consistent, frequent REORG of indexes or table spaces becomes necessary.

Pseudo delete index cleanup

DB2 11 reduces the need to REORG indexes to maintain consistent index access performance by providing an automatic clean-up of the pseudo deleted entries in index pages. A 5-day long performance measurement using the IBM WebSphere Portal workload shows 20% CPU reduction by cleaning up the impact of pseudo deleted index entries. This reduces the need for REORG index due to pseudo deleted entries. The clean-up process runs as default in DB2 11 CM under the system service tasks. These tasks are eligible for zIIP processing.

Reduction of indirect references

Indirect references, or overflow records, are created when there is no more room in the data pages (often happening with variable length fields or compressed records) and impacts clustering and increases the cost of accessing the rows. DB2 11 reduces the indirect references created during the update process. This can be done through autonomic monitoring by turning on the option through a new system parameter in NFM.

1.3.6 Application performance

Not only general optimization but also application level performance enhancements are added in DB2 11. They typically require application changes or DDL changes. These are not considered as “out of the box” CPU reduction features but can provide significant performance improvements to the applications, which can enable the new features.

DGTT improvement

Applications using declared global temporary tables (DGTTs) can use the NOT LOGGED option starting in DB2 11 NFM. With NOT LOGGED, the applications updating DGTTs can see CPU and elapsed time reduction by avoiding logging for DGTTs.

DB2 11 also provides a way to avoid repeated full prepare or incremental REBIND of the SQL referencing DGTTs across COMMITs.

EXCLUDE NULL index support

A new index option, EXCLUDE NULL KEYS, is effectively reducing index maintenance cost if the index contains a large number of NULL values. The CPU saving can be seen in INSERT, DELETE, CREATE INDEX or utility operations against the indexes.

Stored procedure improvements

Stored procedures called remotely and using the AUTO COMMIT option can take advantage of implicit commit features in DB2 11.

ARRAY data type is introduced for SQL procedures, which can reduce the application development cost, as well as CPU impact, of handling the array of input data to SQL procedures.

Improved monitoring of stored procedures offers better monitoring granularity allowing the identification of critical statements.

pureXML enhancements

Numerous performance enhancements are provided in XML processing in DB2 11. Some of them are retrofitted to DB2 10. They include optimization of transaction processing, XMLTABLE, XQuery, and XML validations enhancements.

1.4 Performance after migration

CPU reductions expectations are based on the assumption that BIND or REBIND is done for plans and packages in DB2 11. There are performance optimizations that do not require REBIND in DB2 11. However, REBIND in DB2 11 is highly recommended. Migrated plans and packages without REBIND in DB2 11 cause additional optimization. For example, existing SELECT procedures (Sprocs) in DB2 10 are disabled until REBIND in DB2 11 is done. Additionally, there are several new optimizations that are triggered by REBIND or REBIND with APREUSE, such as new column level procedures or decompression improvement.

In addition to REBIND, there are several other factors that can influence performance and we look at them using the migration measurements from a few sample workloads. This section describes these measurements.

1.4.1 Sample OLTP workloads: Migration scenario

This section describes the OLTP workloads performance measurement results for the following workload types:

- IBM Relational Warehouse workload
- IBM Brokerage transaction workload

IBM Relational Warehouse workload

IBM Relational Warehouse workload (IRWW) executes relatively simple IMS transactions under a data sharing environment. Figure 1-3 shows similar migration performance measurements using IRWW.

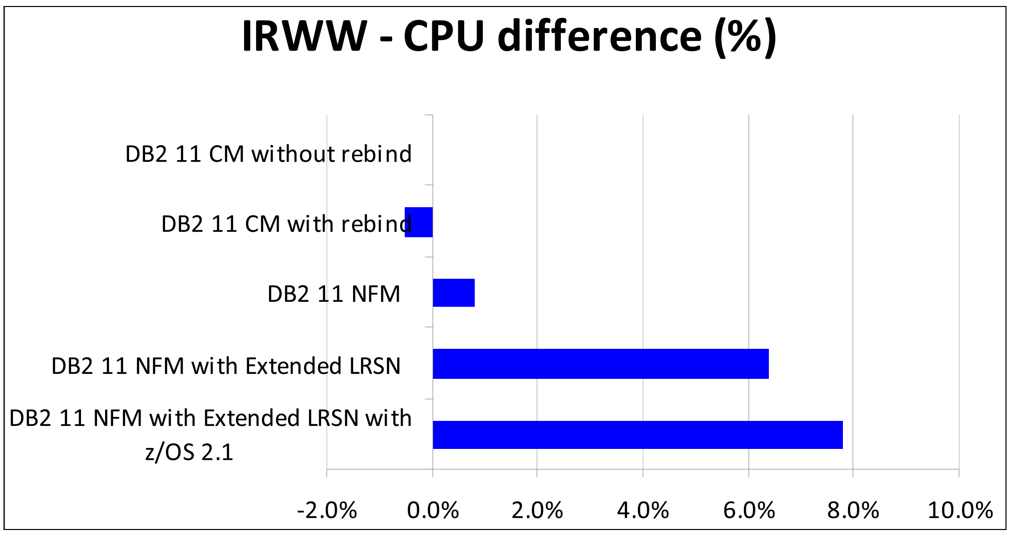

Figure 1-3 IRWW Workload migration scenario

The base measurement is taken under DB2 10 NFM and z/OS V1.13. Since the transactions executed in this workload are simple and short running, the margin of variability is around 1% (plus or minus). Unlike the IBM Brokerage transaction workload, IRWW does not benefit from DB2 11 strong performance points. As a result, DB2 11 measurements show equivalent performance with the base DB2 10 measurement with or without REBIND, or in NFM using basic LRSN.

However, when the log LRSN is converted to extended format, the workload shows more than 6% CPU reduction. This is because one of transactions executes sequential delete (10 deletes per transaction) and the transaction benefits from LRSN spin avoidance.

When the workload is executed under z/OS V2.1, there is additional 1% CPU reduction due to DFSMS CPU reduction from I/Os. This improvement should be seen regardless of the DB2 version. Flash Express is not defined in this system.

IBM Brokerage transaction workload

The IBM Brokerage transaction workload is a complex OLTP workload, which executes various stored procedures to simulate transactions in a brokerage firm.

The base measurements were taken under these conditions:

- DB2 10 NFM
- z/OS V1.13

- One-way data sharing
- zEC12 LPAR with 16 processors defined

Figure 1-4 shows the performance measurements by using the IBM Brokerage online transaction workload.

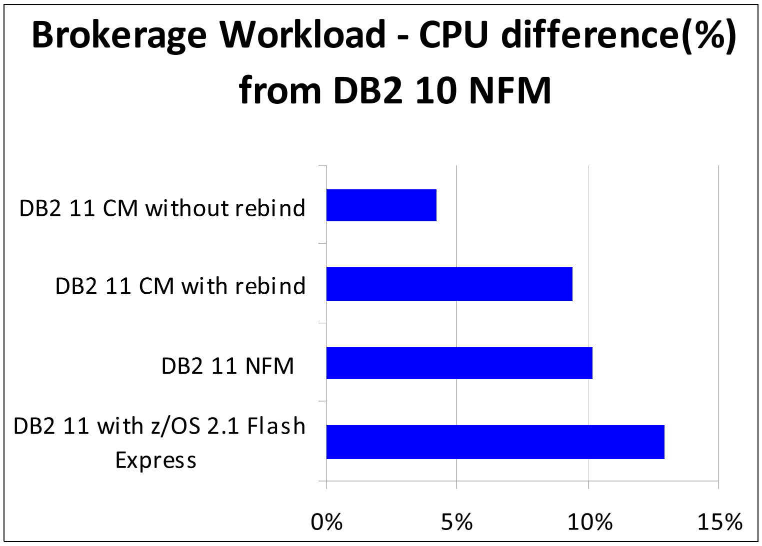

Figure 1-4 Brokerage workload migration scenario

The workload executes 30 SQL procedures and achieves 5000 - 6000 transactions per second. We look at CPU reduction rate compared to the base with the various stages in the migration to DB2 11:

- After migration to DB2 11 CM without executing REBIND of the stored procedure packages, we see a 4% CPU reduction compared to the DB2 10 base test.
- Then, REBIND is done using APREUSE to ensure the same access path. CPU reduction goes up to 9.4%.
- Conversion to NFM shows a similar improvement of 10%.
- With the upgrade of the operating system from z/OS V1.13 to V2.1 and configuring Flash Express, the CPU reduction goes up to 13% compared to the base DB2 10 execution under z/OS V1.13.

There are two reasons why there is additional improvement with z/OS V2.1:

- CPU reduction from I/Os in DFSMS provided by z/OS V2.1. This should be applicable to any DB2 version.
- With Flash Express installed under z/OS V2.1, DB2 11 can allocate execution code in 1 MB pageable frames. This can reduce the translation lookaside buffer misses, which translate as a few % of CPU reduction in this workload.

This workload happens to hit multiple DB2 11 enhancements such as the one related to a large number of partitions (several table spaces are defined as 1000 parts table spaces) and to the support of large buffer pools. As a result, DB2 11 shows close to 10% CPU reduction compared to DB2 10 after REBIND is done.

1.4.2 Sample query workload: Migration scenario

The IBM Relational Warehouse OLTP and the IBM Brokerage transaction workloads show no or minor access path selection differences between DB2 10 and DB2 11. This section describes the impacts of access path selection using a query workloads.

Query workload

This workload executes 22 queries from a static application. Figure 1-5 summarizes the performance measurements by using the query workload.

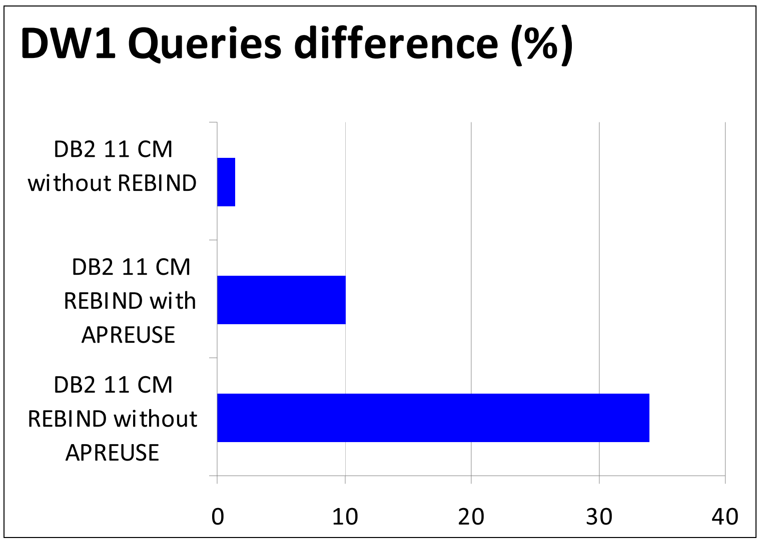

Figure 1-5 Query workload CPU migration scenario

Most of large tables are compressed in this workload. The base measurement is taken under DB2 10 NFM and z/OS V1.13.

With the migration to DB2 11 CM, without REBIND, we see a very small difference from DB2 10. Then REBIND is done to obtain the same access path using APREUSE(ERROR) parameter. In these conditions, 10% CPU reduction from DB2 10 is observed as an average from the 22 queries.

This demonstrates the benefit from enhancements without access path changes in query workloads. They include decompression improvement, sort CPU reduction. Once REBIND without APREUSE is done, an impressive average 34% CPU reduction is observed. The major contribution here is the optimizer cost estimation improvement and wider sparse index usage in DB2 11.

1.4.3 Conclusions

DB2 11 performance improvement is heavily influenced by various factors, such as REBIND, access path selection changes, LOG RBA/LRSN conversion, or z/OS features. The measurements also demonstrate the wide range of differences in improvement depending whether the workload includes the strong point of DB2 11.

Figure 1-6 shows a list of the major DB2 11 performance functions specifying if REBIND is required. A more detailed list in provided in Appendix C, “To REBIND or not to REBIND” on page 301.

Which Items require REBIND to see benefit?

DB2 11 System Performance Optimization/Features	Triggering action
General code optimizations	None
Customized code generation for sort	None
Optimizing the access against the table with large number of partitions	None
REL(DEALLOCATE) optimization	None
DB2 latch reductions (LC14)	None
Buffer Pool enhancements	None
More zIIP processing	None
DDF Sync Receive enhancements	TCP/IP APAR
Data sharing performance improvement	None
Customized code generation for column processing	REBIND (OK with APREUSE)
Decompression Improvement	REBIND (OK with APREUSE)
xProc above the bar	REBIND (OK with APREUSE)

Figure 1-6 Required actions for some DB2 11 system performance functions

1.5 How to read and use this book

This IBM Redbooks publication is a technical report of detailed performance measurements of most of the performance-related topics of DB2 11 for z/OS. The measurements are meant to show how the new functionalities perform by comparing them to the same functionalities executed in DB2 10, if existing, or to analogous similar solutions.

We also describe some usage considerations as derived from the measurements with the intent to provide an understanding of the options of the function and highlight reasonable expectation from running it.

Most measurements were done in a laboratory environment, and as such, they have been run in a dedicated environment and focused on a specific function, being atypical for the absence of other workloads causing resource contention. Extrapolation and generalization require judgment.

This book does not cover general DB2 monitoring nor DB2 11 general functions. For such topics, refer to *Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS*, SG24-8182 and *DB2 11 Technical Overview*, SG24-8180.

1.5.1 Book's structure

Here is a quick roadmap to this book's structure:

- Chapter 1, "Introduction" on page 1

  This chapter presents the executive summary, including general performance expectations in DB2 for z/OS 11

- Chapter 2, "Subsystem" on page 21

  Most of the many new DB2 engine functions are described here. Most of the enhancements do not require any manual intervention.

- Chapter 3, "Synergy with z platform" on page 65

  This can be considered together with the Subsystem chapter, but this chapter focuses the features that take advantages of System z environment, hardware or software.

- Chapter 4, "Availability" on page 81

  This chapter deals with functions that reduce the need to run REORG.

- Chapter 5, "IBM DB2 Analytics Accelerator" on page 95

  This chapter discusses the improvements in IBM DB2 Analytics Accelerator functions and performance.

- Chapter 6, "Data sharing" on page 111

  This chapter describes data sharing related enhancements.

- Chapter 7, "SQL" on page 125

  Here are described several enhancements in the access path selection, transparent to the application developer. The description includes the access path stability items as well.

- Chapter 8, "Application enablement" on page 165

  This chapter is all about new application functions, the use of which substantially improves the productivity of application developers. It includes various new functions to support big data and JSON support.

- Chapter 9, "Distributed environment" on page 211

  This chapter includes functions that in other times were called e-business, focusing on the performance enhancements in remote access.

- Chapter 10, "Utilities" on page 229

  Utilities continue to improve its performance, especially in REORG, RUNSTATS, and LOAD and RECOVER.

- Chapter 11, "Installation and migration" on page 245

  This chapter provides measurements of migration and skip-migration steps for sample DB2 catalogs.

- Chapter 12, "Monitoring" on page 267

  Here we show how DB2 11 has enhanced instrumentation reducing NOT ACCOUNT time.

- Appendix A, "Recent maintenance" on page 289

  This appendix is an attempt at keeping up with what the maintenance stream is adding in terms of functions and performance after the general availability of DB2 11.

- Appendix B, "The IBM DB2 workloads" on page 297

  This appendix summarizes the characteristics of the workloads mentioned during the definition of the measurements.

- Appendix C, “To REBIND or not to REBIND” on page 301

  To help you plan for REBIND, Appendix C summarizes the DB2 11 functions trying to clarify if they are available in NFM or CM, and if rebinding or other requisite actions are needed.

1.5.2 Measurement environments

Unless described otherwise, the following measurements environment is generally used to obtain the results of the base cases described in this book. BIND, REBIND, or REBIND with APREUSE is done for the static plans and packages executed under DB2 11:

- System z EC12 dedicated z/OS and ICF LPARs
- Dedicated network
- z/OS 1.13
- Dedicated IBM DS8000® family with zHPF enabled

2

Subsystem

DB2 evolves with version 11 removing structural constraints to support its increasing use by concurrent workloads, and to enhance its availability. Several improvements to the DB2 engine allow faster access for applications and easier database administration.

In this chapter, we describe the following topics:

- Extended log RBA and LRSN
- Virtual storage enhancements
- CPU reductions with selective decompression
- Column processing runtime optimization
- Optimization of RELEASE(DEALLOCATE) BIND/REBIND option
- Buffer pool management
- Change Data Capture Log read (IFCID 306) improvement
- ACCESS DB command performance improvement
- Security
- Scalability improvements with many partitions
- Persistent thread break-in
- Classic IRWW workload measurements
- High insert batch workload measurements
- SAP Banking Service Day Posting workload measurements

2.1 Extended log RBA and LRSN

The DB2 log relative byte address (RBA) identifies the position in the DB2 log used for recovery. The original design of DB2 used a 6-byte value for the RBA and limited the total capacity of the DB2 log to 256 TB. As hardware performance and the log generating speed of DB2 have increased with workloads, the capacity limit of the log is quickly being reached. When the limit is reached, a disruptive process must be followed to set the RBA to 0 to effectively reset the log capacity back to the maximum.

In data sharing, the 6-byte log record sequence number (LRSN) also had limits. The 6-byte LRSN is derived from the 8-byte TOD clock to generate a unique sequence value for all of the logs across the data sharing system. While the TOD clock on which the LRSN is based does not reach a limit until 2042, some instances of actions such as disabling and re-enabling data sharing may have caused the LRSN to be based on a “future” time. For these systems, 2042 for LRSN purposes might come much sooner.

In addition to DB2 log capacity, the 6-byte LRSN generation was often insufficient to quickly achieve the uniqueness of the 6-byte value that is required. Since the TOD clock conversion results in an equivalent LRSN precision of 16 microseconds, the DB2 Log Manager might need to *spin* (that is, wait for a new TOD value) to obtain the unique LRSN value. Generating uniqueness is a relatively new problem brought on by faster CPU speeds. Changes in recent DB2 releases have reduced the need for the LRSN spin, but it is still required in some cases.

To address all of these issues, DB2 11 implements a 10-byte RBA and LRSN. This greatly increases the RBA and LRSN limits and eliminates the LRSN spin issue.

The 10-byte RBA allows addressing capacity of 1 yottabyte (2**80). The 10-byte LRSN supports 30,000 years and 16 million times more precision.

> **Note:** Refer to Chapter 3, “Scalability” in the IBM Redbooks publication *DB2 11 for z/OS Technical Overview*, SG24-8180, for background details.

2.1.1 Extended RBA in general

The existing 6-byte DB2 RBA and LRSN values exist pervasively in fixed areas of data structures within DB2. Conversion of these data structures is necessary before the full (but optional) implementation of extended RBA function can be accomplished in new function mode (NFM). If the optional conversion steps are not run in NFM, DB2 11 continues to generate log records in the old format with the previous limited capacity.

To fully implement the extended RBA capability in NFM, the optional steps of bootstrap data set (BSDS) conversion and use of an extended RBA/LRSN format in all DB2 objects are required. The extended format change of the RBA in the pages can occur either by utility or when the objects are created.

The migration steps can be performed in any order and are detailed in Chapter 12, “Installation and migration” of the IBM Redbooks publication, *DB2 11 for z/OS Technical Overview*, SG24-8180.

An overview of the situation in DB2 conversion mode (CM) and the following steps in NFM are presented in Figure 2-1 on page 23.

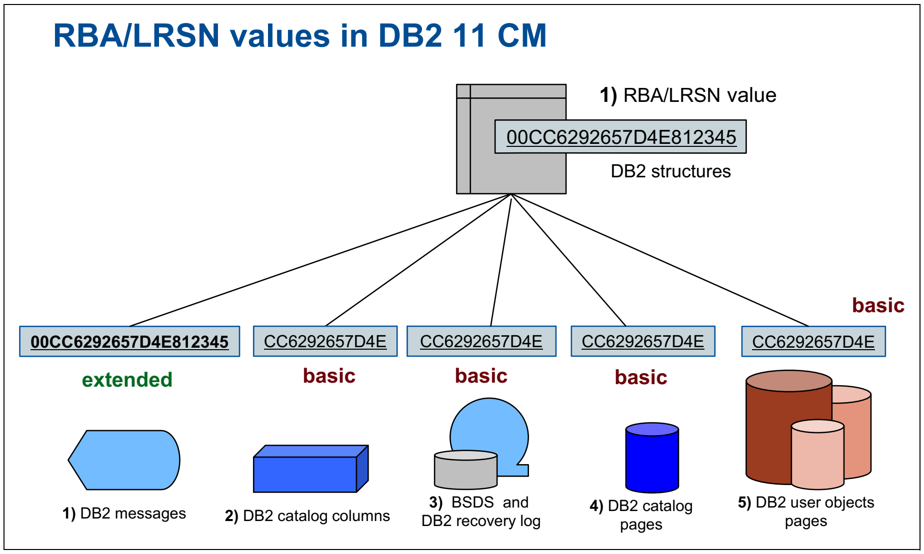

Figure 2-1 Extended RBA/LRSN: The starting point in DB2 11 CM

Here is a logical (but not mandatory) sequence of steps for migrating all DB2 objects to the extended RBA/LRSN format[1]:

- DB2 11 CM is already using the extended RBA/LRSN format in its code, its structures, and messages, noted as **1)** in the figure.
- When you migrate to DB2 NFM (job DSNTIJNF), the contents of DB2 catalog columns noted as **2)** start to reflect the extended format in the catalog contents.
- When you are in NFM and run the new installation job DSNTIJCB, you expand the BSDS to an 8 KB control interval size, and BSDS and log use the extended format, noted as **3)**. This is a step that requires DB2 to be stopped. It would seem reasonable to plan for it together with enabling-new-function mode (ENFM) activity.

  You can now set the OBJECT_CREATE_FORMAT subsystem parameter to EXTENDED to let DB2 create all new table spaces and indexes with extended RBA/LRSN format.

  You can also set the value of the UTILITY_OBJECT_CONVERSION subsystem parameter to NOBASIC to allow DB2 utilities to accept only the RBALRSN_CONVERSION option to convert existing table spaces and indexes to a 10-byte page format.
- You can run the new job DSNTIJCV job to convert the catalog and directory to extended RBA/LRSN format, noted as **4)**. It is a REORG SHRLEVEL CHANGE, which can be logically interrupted and does not require shutdown.
- You now prioritize your databases, possibly by logging intensity, and convert all user data (noted as **5)**) across a reasonable time by running the utilities.
  The RBALRSN_CONVERSION keyword is available for the following utility control statements if UTILITY_OBJECT_CONVERSION is not set to NOBASIC:
  - `REORG TABLESPACE`
  - `REORG INDEX`
  - `REBUILD INDEX`
  - `LOAD`

  Take an image copy after the REORG SHRLEVEL CHANGE/REFERENCE used to convert.

1 With a new installation of DB2 11, all objects are created in extended mode but the BSDS conversion job DSNTIJCB still needs to be run.

RECOVER accepts any format and can convert back to BASIC if so required by choosing a PIT before the conversion.

2.1.2 Impact on performance

In CM or in NFM, even if the necessary conversion steps have not been done, DB2 11 always uses the new 10-byte RBA values in its code and convert these to the old 6-byte format for log records. This is necessary to allow fallback or data sharing coexistence with DB2 10. This 10 to 6-byte conversion to the old log format conversion adds a small CPU overhead to intensive logging activity that occurs with concentrated INSERT/UPDATE/DELETE (I/U/D) activity, for instance. However, since this overhead cannot be isolated from other DB2 11 changes that otherwise improve SQL and LOG performance, the overall performance impact compared to DB2 10 is likely to be in the low single digit percentage range but is not explicitly measurable.

In this section, we show the INSERT, UPDATE, and DELETE measurements at the different stages and look at the results in two cases: small (large percentage of changed data) and large (small percentage of changed data) tables.

INSERT, UPDATE, and DELETE for a small table

We compare a simple batch execution with a small table definition of 2 M INSERT, UPDATE, and DELETE at various DB2 configuration levels.

Figure 2-2 shows the CPU results when running this non-data sharing batch (single thread) INSERT, UPDATE, and DELETE for a small table (row of about 120 bytes with few and short columns).

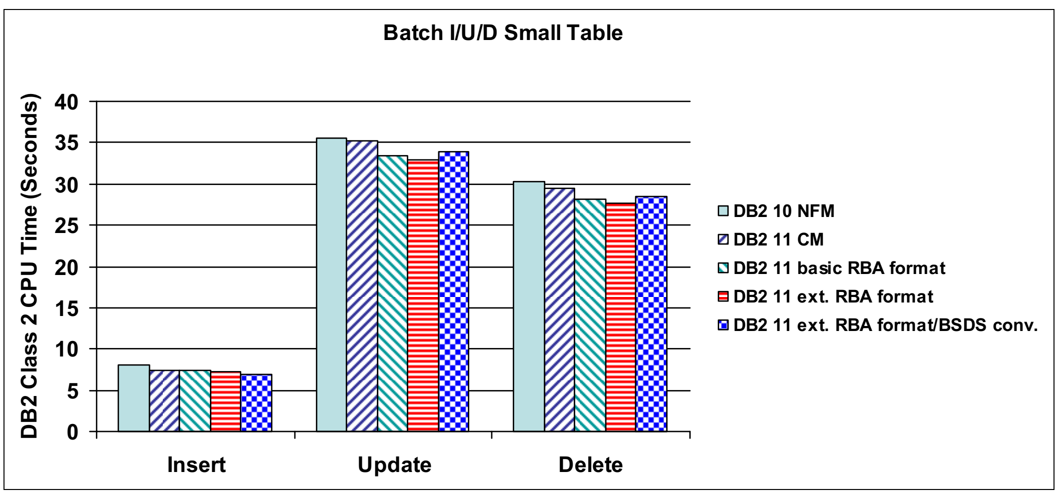

Figure 2-2 Batch execution with a small table definition of 2 M INSERT, UPDATE, and DELETE at various DB2 configuration levels

Overall, the performance is improved compared to DB2 10. The rightmost I/U/D test bars are with DB2 11 with the BSDS converted and extended RBA/LRSN format, in NFM, of the data allowing the utilization of the extended RBA format for the log records.

The functional difference in DB2 11 between the last two tests on the right is only the BSDS conversion so it more likely shows the isolated effect of just using extended RBA without the conversion to the old RBA and log record format.

The last level of DB2 11, having the BSDS converted and using extended RBA/LRSN format, shows INSERT having 3.3% lower class 2 CPU versus DB2 11 without the BSDS converted while UPDATE and DELETE show similarly slightly higher class 2 CPU versus DB2 11 without the BSDS converted of 2.9% for UPDATE, and 3.4% for DELETE. Compared to DB2 10, the I/U/D show 13.8% for INSERT, 4.7% for UPDATE, and 6.0% for DELETE reduced class 2 CPU.

Figure 2-3 shows the average size in bytes of the log records for each individual test with small tables. The last level of DB2 11 (with the BSDS converted and using extended RBA/LRSN format) shows a range 34 - 42% increase in the size of average log records written. The increase is about 50 bytes in the average log record data written when the BSDS is converted. The log records include fixed data areas that must accommodate the size of the increased RBA size so smaller logged data (for a 150-byte row) may have a relatively large increase in log record data volume.

This likely contributes to some of the difference in performance.

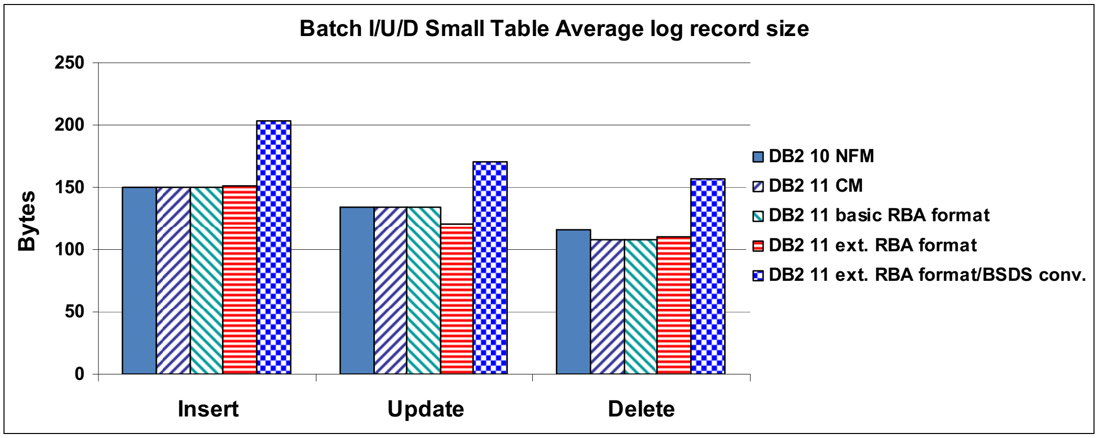

Figure 2-3 Average size in bytes of the log records for each individual test

> **Important:** After conversion to extended RBA/LRSN, allow for log allocation growth with update intensive activity on narrow tables.

INSERT, UPDATE, and DELETE for a large table

Figure 2-4 on page 26 shows sample results when running non-data sharing batch (single thread) of 2 M INSERT, UPDATE, and DELETE for a larger table definition (that has more and larger columns than the previous example and hence larger logging records). Overall, the performance is also improved compared to DB2 10, like the smaller table test above. The last test at DB2 11 with the BSDS converted and extended RBA/LRSN format allows utilization of the extended RBA format for the log records.

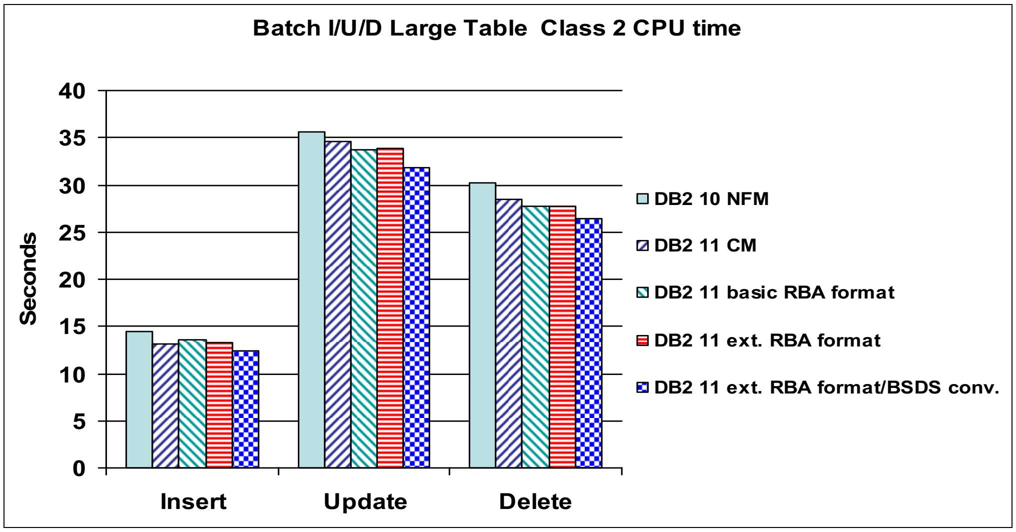

Figure 2-4 Batch execution with a large table of 2 M INSERT, UPDATE, and DELETE SQL each at various DB2 configuration levels with DB2 non-data sharing

The difference in DB2 11 between the last two tests is only the BSDS conversion so it more likely shows the effect of just using extended RBA without the conversion back to the old format log records and 6-byte RBA.

With DB2 11 NFM having the BSDS converted and using extended RBA/LRSN allowing use of extended RBA, shows INSERT, UPDATE, DELETE having 6.8%, 5.9%, and 4.6% lower class 2 CPU versus DB2 11 without the BSDS converted. Compared to DB2 10, the I/U/D show 14.1%, 10.1%, and 12.8% reduced class 2 CPU.

Figure 2-5 for the same test shows the average size of the log records written. The increase (3 to 5%) in the average log record written is much smaller than the previous test as the fixed overhead areas associated explicitly with the larger 10-byte RBA data are not as significant with individual larger logged data sizes.

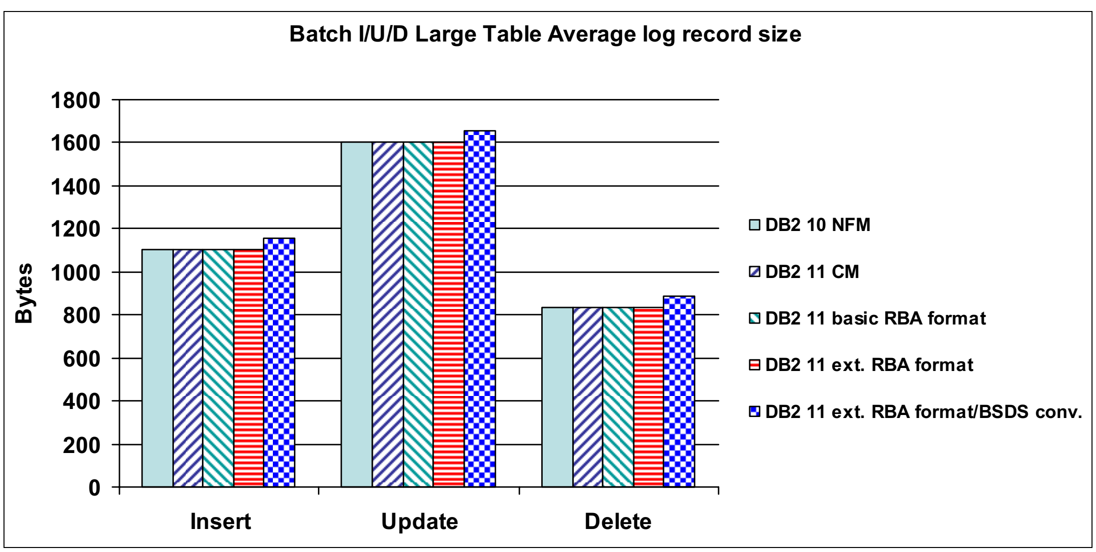

Figure 2-5 Average size in bytes of the log records: Large table

2.1.3 LRSN spin avoidance

In data sharing, the granularity of the 6-byte LRSN is equivalent to 16 microseconds from the TOD clock. When consecutive log records need a unique LRSN, a CPU spin is used to get a unique value. This can have a significant impact on performance. The expansion of the LRSN to 10 bytes completely eliminates the need for this spin as the granularity is increased to the full extent of the 8-byte TOD clock. (1 picosecond)

After migrating to NFM, optional steps of converting the BSDS and converting to extended RBA/LRSN format allow the elimination of the LRSN spin.

To demonstrate the effect of the LRSN spin avoidance on commonly used SQL, the results of an application doing 2 million INSERT, UPDATE, and DELETE SQL executions are shown in Figure 2-6. The test is run as a single batch thread on an EC12 showing a comparison of the CPU time of LRSN spin events for each SQL type and DB2 level.

Figure 2-6 shows how extended RBA/LRSN with spin elimination can improve SQL performance. For this test, DB2 11 NFM with the new BSDS and extended RBA/LRSN format, a single thread stream of INSERT improved by 25.4% class 2 CPU versus DB2 11 CM, UPDATE by 47.8%, and DELETE by 14.2%.

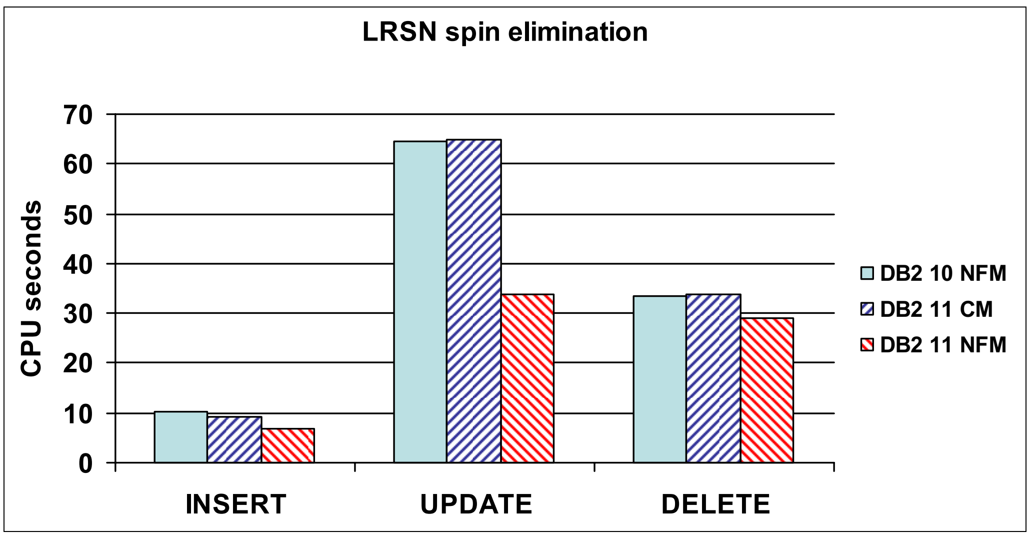

Figure 2-6 LRSN spin elimination

Details about using utilities for converting page sets to use the extended RBA/LRSN format can be found in Chapter 12, “Installation and migration” of the IBM Redbooks publication *DB2 11 for z/OS Technical Overview*, SG24-8180.

Not all of your page sets or partitions need to be REORGed at once to obtain large benefits. Consider first REORGing the page sets or partitions that are updated the most.

2.2 Virtual storage enhancements

In this section, we describe the changes that affect virtual storage usage involving 31-bit common system area (CSA) storage, 64-bit high common storage area (HCSA)[2] storage, as well as 31-bit and 64-bit private DBM1 storage.

[2] HCSA is defined by the IEASYSxx HVCOMMON. This 64-bit common area is placed below the 4 TB line. You can check it with D VS,HVCOMMON.

No specific measurements of real storage usage verification were implemented with DB2 11. During the measurements, we observed an increase in real storage use in the order of 5 to 10%, in line with the growth from earlier versions. The recommendation remains to avoid paging for DB2 pools, even with Flash Express available.

We examine the following topics:

- Moving log buffers to HCSA with PC=NO
- Common storage above the bar
- xProc storage

2.2.1 Moving log buffers to HCSA with PC=NO

In prior DB2 versions, because log buffers reside in the MSTR address space, the interface to log record write is through a space switching program call space switch (PC-SS) instruction, which must first switch to the MSTR address space. With the change in DB2 11, log buffers are moved to 64-bit HCSA in the DBM1 address space and the interface to log record write is branch entered to avoid address space switching during writing log records. There is no need to invoke space switching program call anymore. Log buffers are page fixed and use 1 MB page frames if available; otherwise, 4 KB page frames are used.

The HCSA is allocated below the 4 TB line in the shared area. See Figure 2-7 for the evolution of address space addressing. Storage Management services and IBM RMF™ support for HCSA provide the infrastructure that is required for users of CSA and extended common system area (ECSA) storage to move data above the bar. z/OS V1.10 moved the IOS UCB extension (UCBX) control blocks to HCSA.

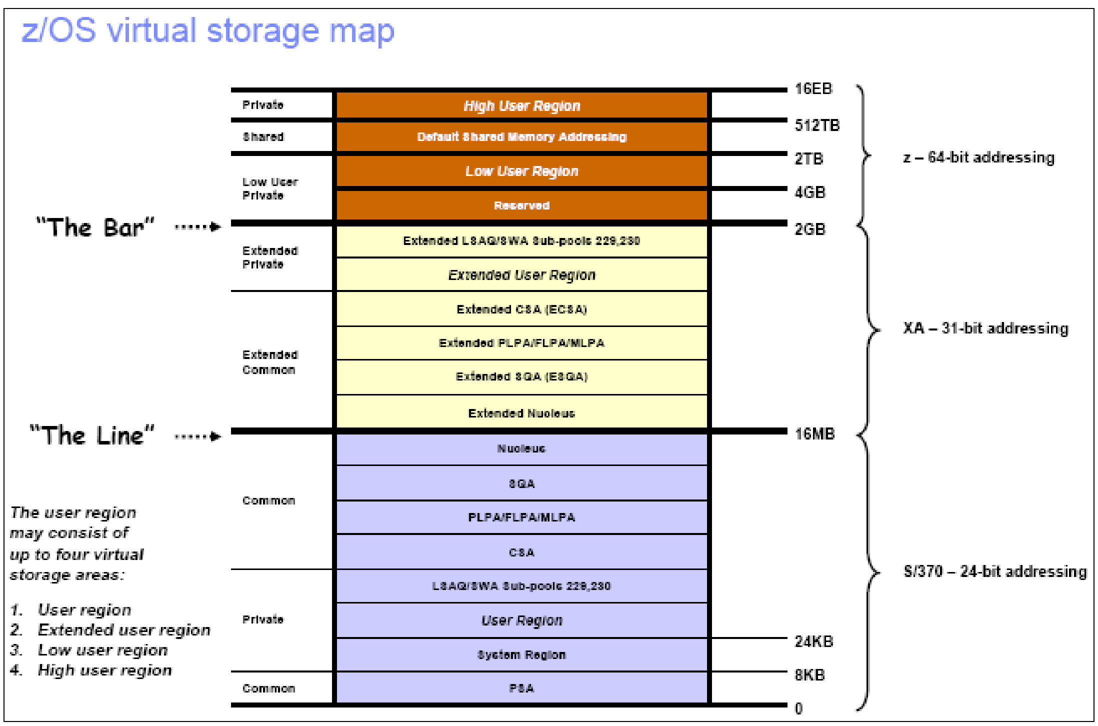

Figure 2-7 Storage map from 24-bit to 64-bit address space

To accommodate this change, users must ensure that their 64-bit HCSA is large enough to accommodate the log buffers (plus about 10% for control structures). HVCOMMON is the system parameter, which is specified in an IEASYSxx parmlib member, that controls the 64-bit common area size from 2 GB to 1 TB. If HVCOMMON is too small for DB2 log manager to allocate its buffers and control blocks, DB2 will not start. Hence, this is a critical migration consideration. The statistics for the common storage used by log manager buffers and control structures can be found in IFCID 225.

Important: Define the IEASYSxx HVCOMMON for the DB2 OUTBUFF+10% to allow DB2 11 to start.

This change in log writing process significantly improves the performance for update intensive online transaction processing (OLTP) and update/insert/delete intensive batch workloads.

Table 2-1 shows the performance results for different OLTP and batch workloads.

Table 2-1 Log enhancement performance results

Workload			
Data sharing		CPU reduction LRSN basic	CPU reduction LRSN extended
	Classic IRWW data sharing	1 to 2%	1 to 2%
	High insert data sharing	5 to 10%	8 to 10%
Non-data sharing	Batch non-data sharing	8 to 14% (insert), 5 to 28% (update), 20% (delete, truncate)	

2.2.2 Common storage above the bar

DB2 11 improves performance by exploiting z/OS functions, reducing path length in several situations, and further reducing virtual storage requirements below-the-bar moving more DB2 structures into 64-bit above-the-bar storage. WWFR[3], CCB[4], and xProcs have been moved to 64-bit address space storage to reduce the cost of moving a section from the EDM Pool or Global Statement Cache to thread storage. This above-the-bar shared memory usage has eliminated most of the data movement and data reformatting between the DBM1 and DIST address spaces, thereby reducing CPU cycles. DB2 also reduces total storage usage by maintaining only one shared copy of the data. These storage improvements allow more threads to be active at the same time, reduce CPU usage for distributed server requests, and reduce ECSA usage by DB2.

A distributed workload was measured on a DB2 11 system. We compared the common storage utilization to the same workload that was run against a DB2 10 subsystem. These DB2 10 and 11 test results were taken from non-data sharing DB2 subsystems. The workload consisted of 500, 1000, and 1500 distributed active connections that were held open before committing. Each held a thread, which enabled a storage usage snapshot to be taken with all the connections and threads open.

Table 2-2 on page 30 shows the amount of common storage that is used below and above the 2 GB bar for both DB2 10 and 11 running with 500, 1000, and 1500 concurrent threads. Both DB2 10 and 11 were in new-function mode. The measurements were taken on a System zEC12 processor running z/OS 1.13.

[3] WWFR is the instrumentation functional request block, where WW is the component code for DB2 IFC.
[4] CCB is the composite capability block containing information related to a DB2 agent.

Table 2-2 Common storage (in megabytes)

Common storage summary (MAX SAMPLES)	DB2 10			DB2 11		
Number of concurrent threads	**500**	**1000**	**1500**	**500**	**1000**	**1500**
Common storage below and above 2 GB						
Extended CSA size	256.75	256.75	256.75	256.75	256.75	256.75
Fixed pool below	4.37	6.11	9.07	3.28	3.34	3.22
Variable pool below	0.89	0.99	1.16	0.84	0.90	0.97
Getmained below	0.08	0.08	0.08	0.10	0.10	0.10
Fixed pool above	5.95	9.38	16.53	7.91	8.36	9.75
Variable pool above	0	0	0	0.98	0.99	0.99
Getmained above	0.19	0.19	0.19	0.19	0.19	0.19
Storage manager control blocks above	1.34	1.34	1.34	1.34	1.34	1.34
Total	269.57	274.84	285.12	271.39	271.97	274.30

Measurements have shown that there is a reduction in common storage in DB2 11:

- Fixed pool below reduction is about 40%
- Variable below reduction is about 5% to 16%
- Fixed pool above reduction is about 11% to 41%

Table 2-3 presents the max samples of the common storage summary from RMF batch VSTOR report.

Table 2-3 Common storage summary

MAX SAMPLES	DB2 10			DB2 11		
Number of active threads	**500**	**1000**	**1500**	**500**	**1000**	**1500**
SQA below 16 MB	1188 KB	1188 KB	1188 KB	1188 KB	1188 KB	1188 KB
CSA below 16 MB	324 KB	324 KB	324 KB	324 KB	324 KB	324 KB
SQA above 16 MB	43.2 MB	44.3 MB	45.8 MB	43.9 MB	44.0 MB	44.1 MB
CSA above 16 MB	63.7 MB	69.4 MB	73.2 MB	63.1 MB	63.3 MB	63.4 MB
Additional per thread - SQA above		2.2 KB	3.0 KB		0.2 KB	0.2 KB
Additional per thread - CSA above		4.2 KB	7.6 KB		0.4 KB	0.2 KB

Measurements showed that the average additional SQA above the bar per thread storage was reduced from 2.2 KB on DB2 10 down to about 0.2 KB on DB2 11, and the average additional CSA above the bar per thread storage was reduced from 4.2 KB on DB2 10 down to 0.4 KB on DB2 11. This is due to the move of the WWFR, CCB, and xProcs to 64-bit storage in the z/OS shared virtual object. With this storage constraint relief, DB2 11 offers again more concurrent threads to be used within a single DB2 subsystem.

2.2.3 xProc storage

The virtual storage footprint of DB2 is affected by various activities. The number and type of SQL statements, whether dynamic or static, may contribute to a big portion of this storage. Each statement being executed by a thread requires its own unique storage allocation. So each instance of an SQL statement by each thread is separately allocated virtual storage.

Before DB2 9, the individual thread SQL statement storage was allocated below the bar, which when aggregated could severely constrain the IBM MVS™ address space 2 GB limit for below the bar storage. DB2 9 implemented changes so that the SQL statement storage would be split between a below the bar portion and an above the bar portion. Through DB2 9, the SQL statement storage would normally persist either for commit or allocation duration depending on BIND parameters or individual statement options. For instance, 100 threads each executing 100 common statements (different statement sections or all open at the same time) in a commit interval could have, at a peak, 10,000 allocated SQL statements (100x100).

DB2 10 made significant changes to move nearly all of the statement storage into above the bar storage. The primary SQL statement storage in DB2 10 that is still allocated below the bar consists of the column processing procedures (SProc, IProc, and so on, commonly referred to as *xProc*). This storage is typically a smaller portion of the entire SQL statement storage. The xProc storage could not be moved to above the bar storage because z/OS *did not support executable code in 64-bit storage*.

In addition, DB2 10 implemented changes so that the xProc storage did not have to be allocated for each thread, but could be shared among all common users of the same statement. One hundred threads each executing 100 common statements would only require the storage for 100 xProcs below the bar (100x1). That means only 1 xProc is needed for all users of the same statement. But the allocation for the above the bar storage for each statement instance would still be 10,000 at a peak.

A limiting factor in DB2 10 for xProc below the bar storage is that if a statement xProc is not in use by a thread, it needs to be freed so that below the bar virtual storage accumulation would not occur for unused statements. So if a common statement is not currently in use, it needs to be allocated again at the next reference. For frequently used statements, the overhead to instantiate the xProc SQL storage is small in copying from a cache and then its initialization.

DB2 11 makes a number of improvements in this SQL statement storage usage that are mostly transparent. The below the bar xProc storage for each statement is now allocated above the bar as z/OS 1.13 supports 64-bit code execution and DB2 11 requires z/OS 1.13 or later. Then, the below the bar virtual storage constraint is no longer an issue for xProc storage, so the above the bar allocation can persist and remain allocated as long as it is in use, or if the skeleton copies remain in the EDM skeleton pool for static statements or the EDM statement cache for dynamically prepared statements. For a very short running SQL statement, this can help performance as the necessary xProc is likely to already exist and does not need initialization if it has recently been used but is not currently in use.

The DB2 11 SQL statement storage changes are effective in CM both for dynamic SQL and for static SQL if the applications are bound in DB2 11. Static SQL statements bound in prior DB2 releases have conversions and expanded storage usage when run in DB2 11 and behave differently.

Now with DB2 11 supporting the xProc storage above the bar and allowing it to persist during the cached statement or package, the xProc storage is effectively cached in a similar manner to the EDM skeleton pool for package static SQL and the EDM statement cache pool for dynamic statements.

DB2 11 includes additional changes affecting SQL statement storage and performance. The SQL statement may have large data structure portions that are fixed (the same) for each thread executing the statement. By moving these fixed areas of SQL statements to the common xProc storage, repeated executions of the same statements by different threads have improved performance by sharing the xProc storage (one copy for each unique statement) while reducing the thread SQL statement storage (one for every thread use of a unique statement).

So in DB2 11, the xProc storage is allocated above the bar, and is larger, but statement execution performs better for repeated statements by multiple threads or by the same thread in different commit intervals for commonly used statements.

For example, the short PREPARE used to instantiate a statement can be more expensive than the statement execution for a very short running SQL statement. By reducing the cost of the *short PREPARE*, overall statement performance is improved.

xProc and thread SQL storage comparison DB2 10 and DB2 11

To demonstrate how the SQL statement storage changes, the following test is created. A total of 40 threads (20 dynamic and 20 static) each executes exactly one different unique statement (no common statements). After the point that each statement is executed, the 40 applications all “pause” so that a STATISTICS TRACE CLASS(1) can capture the state of the statement storage with IFCIDs 2 and 225. The pause is necessary to hold the storage footprint for the statistics interval (or IFI READS) to get a snapshot. If the application ends, or a COMMIT occurs, the storage could be freed if there are no other users of the xProc statement storage.

In DB2 11, the xProc storage remains allocated but the thread SQL storage (THREAD PLAN AND PACKAGE STORAGE) is freed after COMMIT.

Figure 2-8 on page 33 shows a comparison of the SQL storage for this test from the statistics report.

The SYSTEM COPIES OF CACHED SQL STMTS is the total storage area for the dynamic SQL xProc. The 'IN USE STORAGE' is the actual storage used in the storage area for the dynamic SQL xProcs. Similarly, the SYSTEM COPIES OF STATIC SQL is the same metric for the static SQL in the applications. The 0.15 MB in DB2 10 moves above the bar in DB2 11 and is eight times larger. The actual statements for dynamic and static are the same so that 0.15 MB storage size being similar between dynamic and static is expected.

The THREAD COPIES OF CACHED SQL STMTS is the aggregate total for the thread SQL dynamic storage. The same metric for static SQL does not exist as it is a part of THREAD PLAN AND PACKAGE STORAGE. Some small variation is expected between DB2 10 and DB2 11 but the demonstration is to show how the storage allocation areas changes.

Figure 2-8 on page 33 shows how the actual “in use” xProc storage moves above the bar and is eight times larger in DB2 11. It also shows how the “in use” portion of the THREAD COPIES OF CACHED SQL STATEMENTS is reduced in size in DB2 11 to account for the portion of the statement storage that is moved into the commonly shared xProc storage area.

```
DB2 10                                                            DB2 11

DBM1 AND MVS STORAGE BELOW 2 GB                          QUANTITY DBM1 AND MVS STORAGE BELOW 2 GB                     QUANTITY
--------------------------------------------  ------------------  --------------------------------------------  ------------------
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  SYSTEM COPIES OF CACHED SQL STMTS      (MB)                1.75   SYSTEM COPIES OF CACHED SQL STMTS      (MB)                N/A
    IN USE STORAGE                       (MB)                0.15     IN USE STORAGE                       (MB)                N/A
    STATEMENTS COUNT                                          N/A     STATEMENTS COUNT                                          N/A
    HWM FOR ALLOCATED STATEMENTS         (MB)                0.15     HWM FOR ALLOCATED STATEMENTS         (MB)                N/A
      STATEMENT COUNT AT HWM                                  N/A       STATEMENT COUNT AT HWM                                  N/A
      DATE AT HWM                                             N/A       DATE AT HWM                                             N/A
      TIME AT HWM                                             N/A       TIME AT HWM                                             N/A
  SYSTEM COPIES OF STATIC SQL            (MB)                0.66   SYSTEM COPIES OF STATIC SQL            (MB)                N/A
    IN USE STORAGE                       (MB)                0.15     IN USE STORAGE                       (MB)                N/A
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

DBM1 STORAGE ABOVE 2 GB                                  QUANTITY DBM1 STORAGE ABOVE 2 GB                             QUANTITY
--------------------------------------------  ------------------  --------------------------------------------  ------------------
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  TOTAL AGENT LOCAL STORAGE              (MB)               90.00   TOTAL AGENT LOCAL STORAGE              (MB)               95.68
    TOTAL AGENT SYSTEM STORAGE           (MB)                1.99     TOTAL AGENT SYSTEM STORAGE           (MB)                3.42
    TOTAL AGENT NON-SYSTEM STORAGE       (MB)               88.01     TOTAL AGENT NON-SYSTEM STORAGE       (MB)               92.26
      THREAD COPIES OF CACHED SQL STMTS  (MB)                 N/A       THREAD COPIES OF CACHED SQL STMTS  (MB)                 N/A
        IN USE STORAGE                   (MB)                6.47         IN USE STORAGE                   (MB)                5.70
        STATEMENTS COUNT                                       20         STATEMENTS COUNT                                       20
        HWM FOR ALLOCATED STATEMENTS     (MB)                6.47         HWM FOR ALLOCATED STATEMENTS     (MB)                5.70
          STATEMENT COUNT AT HWM                               20           STATEMENT COUNT AT HWM                               20
          DATE AT HWM                                    11/16/13           DATE AT HWM                                    11/16/13
          TIME AT HWM                                 01:28:45.74           TIME AT HWM                                 01:33:56.62
  DYNAMIC STMT CACHE CNTL BLKS           (MB)                0.55   DYNAMIC STMT CACHE CNTL BLKS           (MB)                0.55
  SYSTEM COPIES OF CACHED SQL STMTS      (MB)                 N/A   SYSTEM COPIES OF CACHED SQL STMTS      (MB)                1.38
    IN USE STORAGE                       (MB)                 N/A     IN USE STORAGE                       (MB)                1.15
    HWM FOR ALLOCATED STATEMENTS         (MB)                 N/A     HWM FOR ALLOCATED STATEMENTS         (MB)                1.15
  SYSTEM COPIES OF STATIC SQL            (MB)                 N/A   SYSTEM COPIES OF STATIC SQL            (MB)                1.43
    IN USE STORAGE                       (MB)                 N/A     IN USE STORAGE                       (MB)                1.05
```

Figure 2-8 Side-by-side comparison of DB2 10 and DB2 11 OMPE storage statistics related to SQL statement storage and xProcs for 40 threads

Figure 2-9 on page 34 shows a similar comparison when running 4000 threads with a similar set of dynamic and static SQL. For the few dynamic SQL statements, the xProc storage increases from 0.15 MB to 1.15 MB as it moves above the bar in DB2 11. Because the few statements are shared by many threads, the IN USE STORAGE of THREAD COPIES OF CACHED SQL STATEMENTS decreases from 645 MB to 568 MB.

This demonstrates how increasing the size of the xProc storage with shareable fixed storage areas of the SQL statement storage can have a much larger reduction in the total thread SQL storage.

```
DB2 10                                                           DB2 11

DBM1 AND MVS STORAGE BELOW 2 GB                        QUANTITY DBM1 AND MVS STORAGE BELOW 2 GB                        QUANTITY
--------------------------------------------  ------------------ --------------------------------------------  ------------------
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  SYSTEM COPIES OF CACHED SQL STMTS      (MB)              1.78   SYSTEM COPIES OF CACHED SQL STMTS      (MB)               N/A
    IN USE STORAGE                       (MB)              0.15     IN USE STORAGE                       (MB)               N/A
    STATEMENTS COUNT                                        N/A     STATEMENTS COUNT                                        N/A
    HWM FOR ALLOCATED STATEMENTS         (MB)              0.15     HWM FOR ALLOCATED STATEMENTS         (MB)               N/A
      STATEMENT COUNT AT HWM                                N/A       STATEMENT COUNT AT HWM                                N/A
      DATE AT HWM                                           N/A       DATE AT HWM                                           N/A
      TIME AT HWM                                           N/A       TIME AT HWM                                           N/A
  SYSTEM COPIES OF STATIC SQL            (MB)              0.73   SYSTEM COPIES OF STATIC SQL            (MB)               N/A
    IN USE STORAGE                       (MB)              0.15     IN USE STORAGE                       (MB)               N/A
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

DBM1 STORAGE ABOVE 2 GB                                QUANTITY DBM1 STORAGE ABOVE 2 GB                                QUANTITY
--------------------------------------------  ------------------ --------------------------------------------  ------------------
  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  TOTAL AGENT LOCAL STORAGE              (MB)           2893.33   TOTAL AGENT LOCAL STORAGE              (MB)           3357.76
    TOTAL AGENT SYSTEM STORAGE           (MB)              2.29     TOTAL AGENT SYSTEM STORAGE           (MB)              3.67
    TOTAL AGENT NON-SYSTEM STORAGE       (MB)           2891.05     TOTAL AGENT NON-SYSTEM STORAGE       (MB)           3354.09
      THREAD COPIES OF CACHED SQL STMTS  (MB)               N/A       THREAD COPIES OF CACHED SQL STMTS  (MB)               N/A
        IN USE STORAGE                   (MB)            645.62         IN USE STORAGE                   (MB)            568.42
        STATEMENTS COUNT                                   2000         STATEMENTS COUNT                                   2000
        HWM FOR ALLOCATED STATEMENTS     (MB)            645.62         HWM FOR ALLOCATED STATEMENTS     (MB)            568.42
          STATEMENT COUNT AT HWM                           2000           STATEMENT COUNT AT HWM                           2000
          DATE AT HWM                                  11/16/13           DATE AT HWM                                  11/16/13
          TIME AT HWM                               00:43:34.26           TIME AT HWM                               00:51:42.26
  DYNAMIC STMT CACHE CNTL BLKS           (MB)              0.55   DYNAMIC STMT CACHE CNTL BLKS           (MB)              0.55
  SYSTEM COPIES OF CACHED SQL STMTS      (MB)               N/A   SYSTEM COPIES OF CACHED SQL STMTS      (MB)              1.38
    IN USE STORAGE                       (MB)               N/A     IN USE STORAGE                       (MB)              1.15
    HWM FOR ALLOCATED STATEMENTS         (MB)               N/A     HWM FOR ALLOCATED STATEMENTS         (MB)              1.15
  SYSTEM COPIES OF STATIC SQL            (MB)               N/A   SYSTEM COPIES OF STATIC SQL            (MB)              1.43
    IN USE STORAGE                       (MB)               N/A     IN USE STORAGE                       (MB)              1.05
```

Figure 2-9 Side-by-side comparison of DB2 10 and DB2 11 OMPE storage statistics related to SQL statement storage and xProcs for 4000 threads

Figure 2-9 shows how the actual IN USE xProc storage moves above the bar and in this example is eight times larger in DB2 11 than DB2 10. It also shows how many threads sharing the same statements reduce the total thread SQL storage in use compared to DB2 10.

SQL statement storage and performance

The DB2 11 SQL statement storage changes improve performance for very short running SQL. The overhead of establishing the statement and xProc can be significant for a short running statement but for a very long running statement would be not noticeable.

There are many factors that affect changes in SQL performance in DB2 11. So isolating only the mentioned SQL statement storage changes and how they affect performance is not precisely measurable. However, the following test helps demonstrate how performance is affected. A very short running simple SQL cursor SELECT is created for this test and nine other variations of the same statement increase the statement size in increasing amounts but not in complexity.

Figure 2-10 on page 35 shows the percentage improvement of the 10 statements versus DB2 10.

One range of tests is for PREPARE only for the dynamic statement. This shows that the performance improvement for this very short running statement is most pronounced for the simplest and smallest statement size. The other range of tests shows the range of performance for PREPARE plus execution (OPEN/FETCH/CLOSE). Similar results occur when comparing static SQL with the same statements or dynamic and static complex SQL.

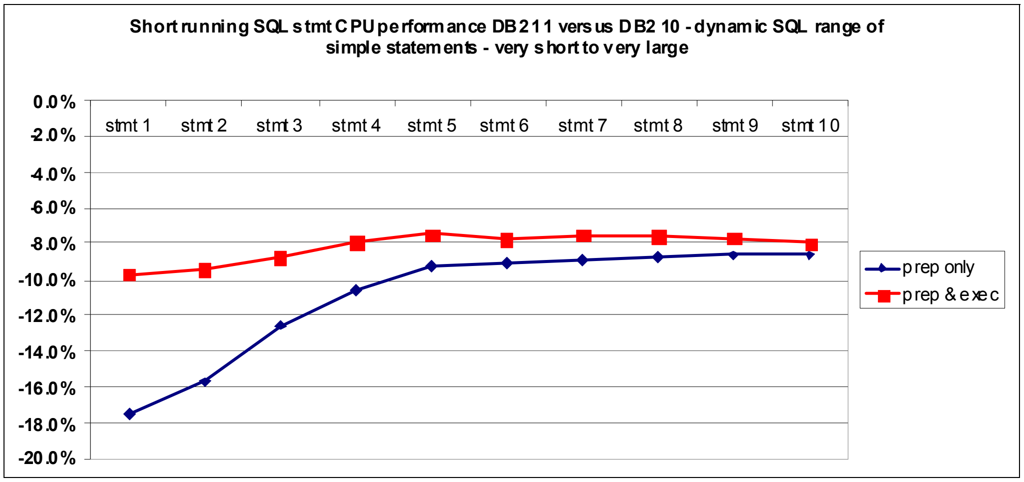

Figure 2-10 Compare DB2 11 versus DB2 10 results of a range of SQL statements that primarily differ in size

The first statement is very short and simple ranging up to the tenth statement that is very long but still simple. One range of tests is for PREPARE only, which stresses the "short prepare" performance. The second range of tests is for the short PREPARE and the statement execution. The performance benefit for the SQL statement storage changes in DB2 11 are small but noticeable for very short running SQL.

2.3 CPU reductions with selective decompression

DB2 uses row-level hardware compression. This requires each row to be decompressed before being passed from the buffer pool to the application for processing. When a query or utility accesses many rows, the CPU time for decompressing rows can be significant. DB2 11 implements some software optimizations to reduce this CPU overhead. The software decompression routine coexists with hardware decompression. The software decompression is in effect without rebind. However, the partial-row decompression optimization is defined at BIND time based on the predicate and select column lists.

This function is effective in DB2 11 CM.

The *partial decompression* requires REBIND, and only applies to reordered row format (RRF) table spaces. Whereas DB2 10 decompresses the entire row regardless of the number of columns referenced by the SQL statement, DB2 11 builds a copy of the expansion dictionary on the first access to an object and only decompresses the portion of the row that is needed starting from the first byte. The columns for predicate evaluation are always decompressed first. Because these columns are filtering rows, any additional columns in the select list are only decompressed if the row qualifies.

The application has to be rebound to make use of all aspects of this enhancement. Users should see substantial reduction in CPU usage for table space scan of very large tables, especially when only a small percentage of the columns is referenced by the SQL.

The following performance measurements demonstrate the decompression optimization. The compressed table had 180 million rows with 16 columns of different type and length.

Position within the row

To demonstrate the CPU reduction from using partial decompression, a sample program executed SQL statements that selected one column from the row. The position of the row varied from column 1 to 16. Each time, a single column is selected and every SQL statement scans the same number of rows.

The query is:

```
SELECT colx FROM tablex where colx = 'N';
```

where x is the position of the column in the row and N does not exist which forces a full table space scan for all instances of x.

Figure 2-11 shows that while the CPU cost of DB2 10 is constant regarding the column position, partial software decompression CPU cost varies with the position of the column accessed. DB2 11 achieves the largest CPU savings for the first column (65%), but even with the last column, DB2 11 reduces the CPU time by 45%.

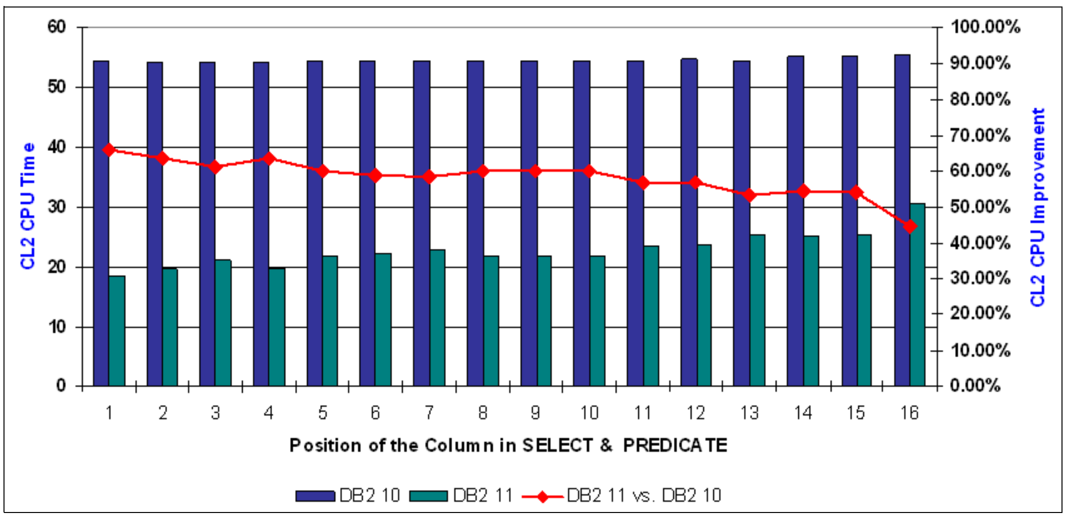

Figure 2-11 DB2 11 versus DB2 10 decompression: Single column selection

Range of columns

Another sample program selected all columns from all rows, a range of columns from all rows or no columns from all rows in a compressed table using table space scan access path.

Figure 2-12 on page 37 shows the results that demonstrated the CPU reduction for decompression with various different types of table scans:

- For the query that does a SELECT COUNT(*), DB2 10 decompressed each row and DB2 11 did not decompress anything, resulting in a 70% CPU reduction.
- The second query, S1-3, retrieved the first three columns. S8-10 retrieved the 8th, 9th, and 10th columns. S13-15 retrieved the 13th, 14th, and 15th columns.
- The last query retrieved every column.

These queries used the same columns selected as predicates (WHERE ... OR) for column ranges (S1-3, S8-10, S13-15) without qualifying any of the rows, which resulted in scanning all the rows in the table for all test cases.

When retrieving any three columns from this table, the CPU savings were 50% to 60%. Even when retrieving every column, the CPU savings were 28%.

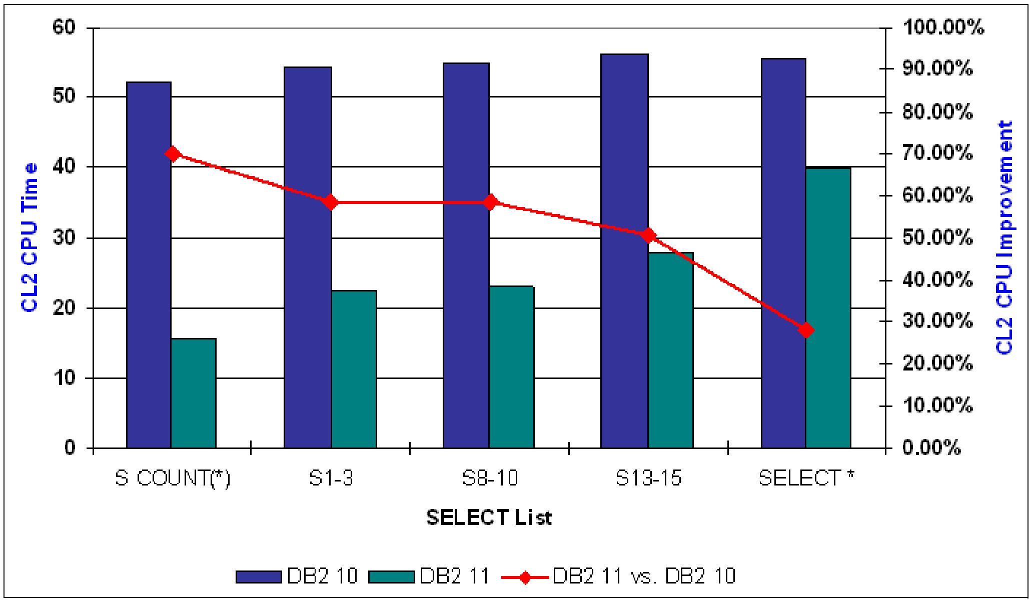

Figure 2-12 CPU reduction for decompression with different types of table scans

Access via index

Index to table access also benefits from this enhancement. This set of test cases used index access to select a couple of columns from ranges of rows in a compressed table.

An example of range access is:

```
SELECT col1, col2 FROM table WHERE ixkey BETWEEN n AND n+range
```

The two range bars on Figure 2-13 on page 38 are a range BETWEEN 9 and 10 M, and a range 9 - 1000.

The random bar is for a random selection of two columns from table by index. An example of SQL used:

```
SELECT col1, col2 FROM table WHERE ixkey = random#
```

In this case only 25% (2.5 M) of rows qualified, although the statement was executed using 10 M different random values.

Figure 2-13 on page 38 demonstrates that the more rows are processed, the larger the class 2 CPU reduction. Even when the index access is random, there is CPU reduction for compressed table.

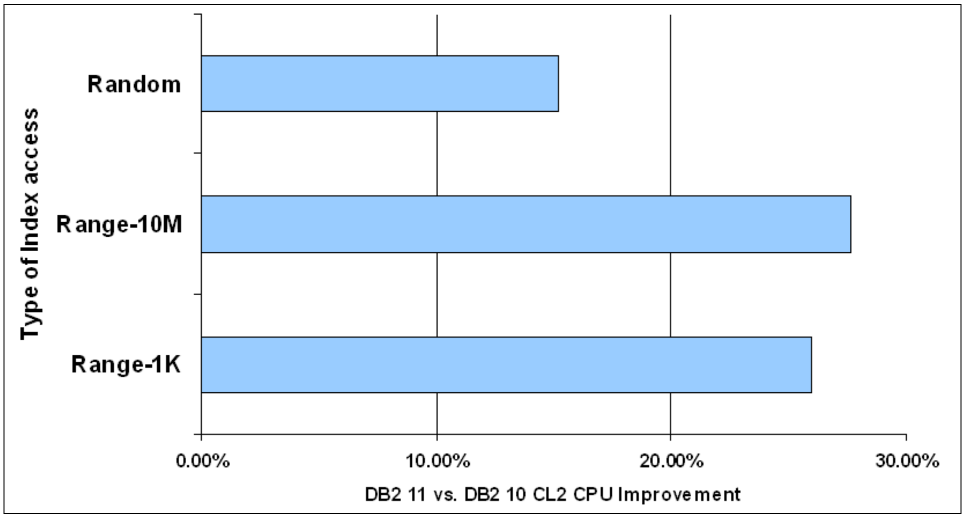

Figure 2-13 Class 2 CPU reduction by index range

The application must be rebound to make use of all aspects of this enhancement. Users should see substantial reduction in CPU usage for table space scan of very large tables especially when only a small percentage of the columns is referenced by the SQL. With predicate filtering, only table rows that qualify are decompressed.

The partial decompression is implicitly used when data is being read from the buffer pool and passed to the application. Most performance workloads use compressed data and automatically benefit from the enhancement.

2.4 Column processing runtime optimization

Various runtime enhancements were done in DB2 11 to improve the performance of input and output column processing.

Input processing refers to INSERT column processing and output refers to SELECT column processing.

2.4.1 Column processing improvements

These improvements are effective in DB2 11 CM when the application is rebound. DECFLOAT data types were especially optimized. Simple fetch processing (SELECT without ORDER BY, GROUP BY, or JOIN) overhead was reduced. There is CPU improvement for SQL statements with CAST or CASE expression. The CAST operator was optimized by generating machine code and CASE expression path length was reduced. Built-in function processing was optimized for the following functions:

- SUBSTR
- DATE
- DIGITS
- COALESCE
- RTRIM
- NULLIF
- CONCAT
- CHAR

Also, some SET statements were optimized.

2.4.2 Performance measurements

Multi-row fetch of a table with 251 columns of various data types showed up to 55% reduction in CPU time in DB2 11. DB2 11 reduced CPU usage for the SAP Day Posting workload by 2%. In general, the CPU improvement is a function of the number of columns selected. More columns yield better performance improvement compared to fewer columns.

Data conversion from and to DECFLOAT showed up to 23% CPU reduction compared to DB2 10. An XML workload that uses DECFLOAT data type for some of its indexes improved in performance by 3.5%.

A single threaded FETCH workload showed up to 5% CPU reduction. Single threaded CASE expressions with a large number of mismatching WHEN clauses have seen up to 20% CPU reduction with optimized code. More complex the CASE expression, better improvement was observed.

There are improvements in both user specified numeric CAST expression as well as DB2 generated CAST expression. More improvement was observed for a SQL statement with more CAST expression. Call to a native SQL procedure to retrieve 500,000 rows in a WHILE loop showed 1.5% CPU reduction with this optimization. At least 3% CPU reduction was observed with built-in function optimization.

In general, column processing of most data types showed substantial performance improvement in CPU usage compared to DB2 10. Figure 2-14 shows the performance results for host variable input columns processing.

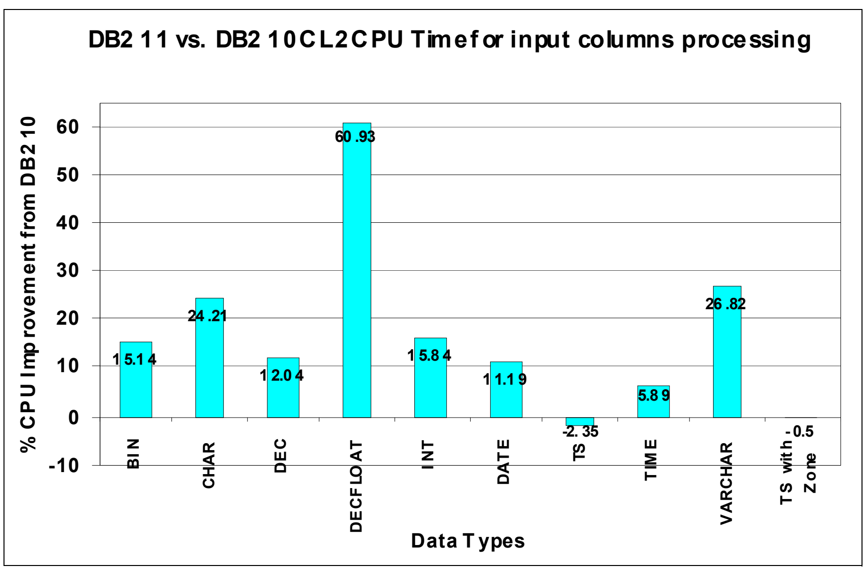

Figure 2-14 CPU usage comparison for host variable input columns processing

Figure 2-15 shows the performance results for host variable output columns processing.

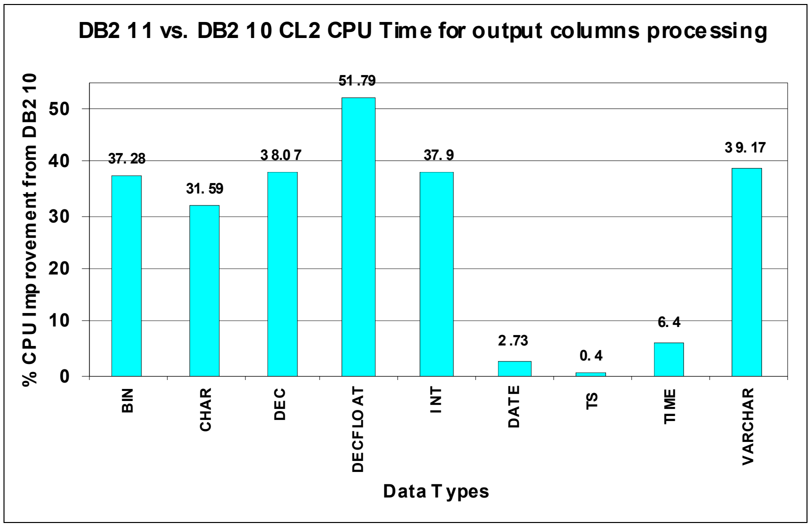

Figure 2-15 CPU usage comparison for host variable output columns processing

Whereas input column processing applies to INSERT statements or host variables used in any SQL predicate, output host variable columns processing are associated with SELECT INTO.

About the negative performance for TS (timestamp), anything within 2% is considered noise level for this type of workload.

2.5 Optimization of RELEASE(DEALLOCATE) BIND/REBIND option

In DB2 10, the RELEASE(DEALLOCATE) option was re-enabled for distributed packages and enhancements were done to use the RELEASE (DEALLOCATE) bind option efficiently, locally or in distributed environments.

The performance improvements in DB2 10 were achieved by avoiding package and section allocations, creation, and clean up of table structure control blocks, and lock/unlock requests for parent locks. This applied for static statements and dynamic statements with KEEPDYNAMIC (YES).

Using this option allowed DBAT threads to hold package allocation locks while they are inactive. All of these enhancements achieved CPU reduction for packages bound with RELEASE(DEALLOCATE) when they contained simple SQL statements with frequent commits. However, packages that used many objects did not benefit from these changes or sometimes even degraded compared to packages bound with RELEASE (COMMIT). To provide consistent performance across different types of workloads, the RELEASE(DEALLOCATE) BIND/REBIND option is optimized in DB2 11.

2.5.1 RELEASE(DEALLOCATE) BIND/REBIND option with DB2 11

In DB2 11, when RELEASE(DEALLOCATE) is used, DB2 monitors the number of locks and table structure blocks held per thread and it releases them at commit once they reach a certain threshold.

The default threshold is high enough that it should not affect existing packages that benefited from the enhancements in DB2 10. This ensures that there is consistent performance improvement, irrespective of how many objects are used or how many locks are held by each package.

Statistics and accounting support is provided by incrementing the counters QXN1093A and QXN1093B marked “serviceability” when respective thresholds are exceeded and a RELEASE (DEALLOCATE) package is released at COMMIT or ROLLBACK.

This function is effective in DB2 11 CM. To enable this behavior, no action is necessary. No REBIND is needed.

2.5.2 Performance measurements

A complex OLTP workload was run that accesses 33 tables with about 15,000 page sets. The Brokerage workload represents transactions of customers dealing in the financial market and updating account information. The transactions are rather complex OLTP transactions implemented in stored procedure using SQL/PL.

These performance measurements demonstrate the CPU reduction for packages that are bound with RELEASE(DEALLOCATE) accessing many objects when compared to packages bound with RELEASE(COMMIT).

Figure 2-16 shows that even when accessing a large number of page sets, a reduction in CPU time was observed for RELEASE(DEALLOCATE) compared to RELEASE(COMMIT).

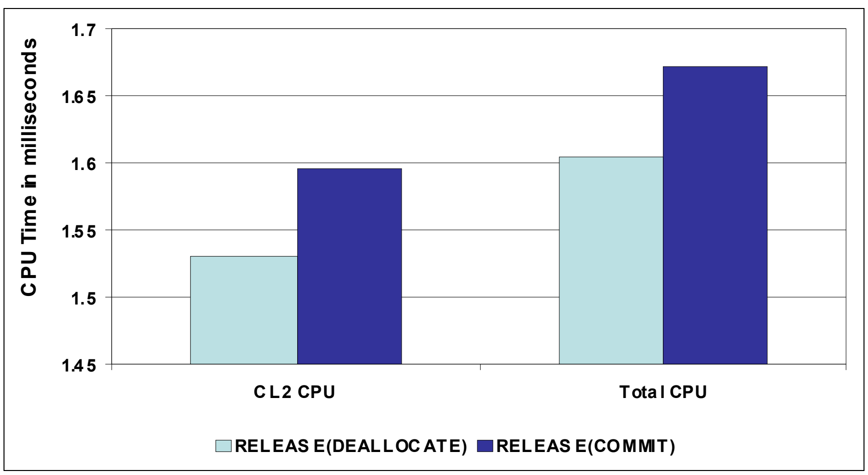

Figure 2-16 RELEASE (DEALLOCATE) versus RELEASE (COMMIT)

2.6 Buffer pool management

In this chapter, we describe many of the changes made to buffer pool management in DB2 11. These topics include:

- Getpage classification
- MRU management
- Sequential detection and dynamic prefetch
- New buffer pool statistics in DB2 11

2.6.1 Getpage classification

Buffer pool tuning is always somewhat of an art, but DB2 11 helps to remove some of the trial-and-error aspect to buffer tuning because the classification of buffers as random versus sequential is better aligned with DB2 prefetch operations, and with sequential format write utility operations. In DB2 11, whenever dynamic prefetch is prefetching pages, the getpages are classified as sequential. Also, whenever a query does a record identifier (RID) list scan (which uses list prefetch), the getpages are classified as sequential. Furthermore, when the utilities are doing sequential format writes, the getpages are classified as sequential. In DB2 10, all of these getpages were classified as random.

When you are interested in understanding the buffer hit ratio or the page residency time, you would like to know the random buffer hit ratio and the residency time for random pages, not the sequential buffer hit ratio or the sequential page residency time. Especially with DB2 11, focus on the random buffer hit ratio or random residency time. The random page residency time can be calculated from the random buffer hit ratio in the following manner. First, you would compute the following two formulas that are shown in Example 2-1.

Example 2-1 Random page residency time calculation formulas

```
VPSIZE / total pages read per second (including pages prefetched)
VPSIZE * (1-VPSEQT/100) / random synchronous I/O/sec
```

The random residency time is the maximum of these two formulas. If the second formula is the larger of the two, it means that the sequential getpages and prefetch operations are affecting the residency time; otherwise they are not.

Avoiding prefetch is generally not a good buffer tuning strategy. Rather, your strategy should be to ensure that there are enough sequential buffers to support the amount of prefetching in your workload. Thus, VPSEQTxVPSIZE needs to be large enough to maximize the sequential prefetch quantity and to support the amount of prefetch parallelism in your workload.

Another problem that is sometimes difficult to detect with DB2 10, is that the buffers containing prefetched pages could be stolen before the getpages that prompted those prefetch operations.

Synchronous I//O for sequential getpage is an indication that the buffers containing prefetched pages are stolen before the getpages that prompted those prefetch operations. However, the sequential synchronous I/O counter was only incremented for sequential prefetch. The statistic was always zero in the case of dynamic and list prefetch. Therefore, the statistic was of limited use. By aligning the getpage classification with prefetch activity, you can count on the sequential synchronous I/O being a precise problem indicator. A few of these I/Os are normal (as it happens that the suspension due to an initial prefetch operation often gets counted as a synchronous I/O), but a large number should be considered abnormal.

Notice that there are some instances of list prefetch that were not changed in DB2 11 to classify the getpages as sequential. Fast lOg Apply and Incremental Copy are examples where the getpages are still classified as random. Only disorganized index scans and RID list scans were changed; however, those are the most common instances of list prefetch in DB2.

The issue is related to how the buffers should be governed. For Incremental Copy, it would be better if the pages were governed by VPSEQT, or even used most recently used (MRU) like Image Copy. Because Fat Log Apply is active by default and is important for recovery performance, it is better to avoid the list prefetch I/O as much as possible if the same page is being updated repetitively.

The READ(NO) getpages that the utilities use to write to a data set sequentially never do any synchronous I/O. In effect, a READ(NO) getpage is one that is acquiring a buffer to create a new page. Since DB2 10 classified such getpages as random, the getpages falsely inflated the random buffer pool hit ratio. By correctly classifying these getpages as sequential, DB2 11 makes the random buffer hit ratio much more accurate. These particular pages use MRU, and have always used MRU, even before DB2 11. That means that once the pages are written to DASD, the pages can be stolen.

There is still one issue with the random buffer hit ratio, and that has to do with sequential inserts. The getpages associated with sequential inserts are classified as random, even when inserts are using READ(NO) getpages. Therefore, the random buffer hit ratio might be falsely inflated, while at the same time any pages that were in the buffer pool before the inserts are being displaced. Therefore, the buffer pool hit ratio as a result of the sequential hit ratio is much worse than it would have been before the inserts. VPSEQT has no effect on the number of buffers consumed by sequential inserts.

2.6.2 MRU management

As stated previously, DB2 uses MRU when utilities write pages sequentially. In Version 9, DB2 also began using MRU when the COPY utility reads pages sequentially, but other utilities such as UNLOAD did not use MRU. UNLOAD classified the getpages as sequential, which meant that lowering VPSEQT would limit buffer pool consumption. Version 11 extends MRU usage to all DB2 utilities, except for LOB table spaces. Consequently, it is not necessary to lower VPSEQT when you want to limit buffer consumption of utilities.

2.6.3 Sequential detection and dynamic prefetch

DB2 10 introduced row level sequential detection (RLSD) for table spaces. Whereas, DB2 9 required a minimum of five getpages to trigger the first dynamic prefetch I/O; RSLD in DB2 10 required a minimum of five rows and two getpages.

It was then detected that triggering prefetch after only two getpages was too aggressive: If a range scan stopped after the second page, DB2 10 was triggering prefetch unnecessarily. It prefetched only eight 4 KB pages, but that was seven pages too much.

DB211 waits one extra getpage. So, DB2 11 requires a minimum of five rows and three getpages. The rationale for this change is based on the idea that the conditional probability of a short transaction continuing to process more rows goes up as it scans more rows. A range scan is likely to process only a subset of the rows in the first page, but it processes all of the rows in the second page. Hence, the conditional probability increases significantly between the second and third pages. Although DB2 11 risks one more synchronous I/O for each range scan, the upside benefit is less unnecessary prefetching.

2.6.4 New buffer pool statistics in DB2 11

DB2 11, and DB2 10 with PM70981, introduced three new statistics. The `DISPLAY BUFFERPOOL` command with the DETAIL option displays them as follows:

- SEQUENTIAL = the length of the SLRU chain.

- VPSEQT HIT = the number of times that the length of the SLRU chain was modified and became exactly equal to VPSEQT.
- RECLASSIFY = the number of times that a buffer hit removed a buffer from the SLRU chain (or, if you are using Version 10, it would have been reclassified if you were using Version 11).

Example 2-2 shows a section of the **-DISPLAY BPOOL(*) DETAIL** command output in DB2 11 highlighting the new statistics.

Example 2-2 Partial output of -DISPLAY BPOOL(*) DETAIL in DB2 11

```
-DISPLAY BPOOL(*) DETAIL
 DSNB401I  -DB1A BUFFERPOOL NAME BP0, BUFFERPOOL ID 0, USE COUNT 55
 DSNB402I  -DB1A BUFFER POOL SIZE = 20000 BUFFERS  AUTOSIZE = NO
            VPSIZE MINIMUM = 0 VPSIZE MAXIMUM = 0
              ALLOCATED       =    20000   TO BE DELETED    =        0
              IN-USE/UPDATED  =       23
 DSNB406I  -DB1A PGFIX ATTRIBUTE -
              CURRENT = NO
              PENDING = NO
            PAGE STEALING METHOD = LRU
 DSNB404I  -DB1A THRESHOLDS -
             VP SEQUENTIAL    = 80
             DEFERRED WRITE   = 30   VERTICAL DEFERRED WRT  =  5,  0
             PARALLEL SEQUENTIAL =50   ASSISTING PARALLEL SEQT=  0
 DSNB546I  -DB1A PREFERRED FRAME SIZE 4K
         20000 BUFFERS USING 4K FRAME SIZE ALLOCATED
 DSNB409I  -DB1A INCREMENTAL STATISTICS SINCE 07:30:00 JAN 15, 2014
 DSNB411I  -DB1A RANDOM GETPAGE    =95
             SYNC READ I/O (R) =0
             SEQ.   GETPAGE    =28
             SYNC READ I/O (S) =0
             DMTH HIT          =0
             PAGE-INS REQUIRED =0
             SEQUENTIAL        =445
             VPSEQT HIT        =0
             RECLASSIFY        =0
 DSNB412I  -DB1A SEQUENTIAL PREFETCH -
              REQUESTS        =0
              PREFETCH I/O    =0
              PAGES READ      =0
...
```

Buffers on the SLRU chain are called *sequential buffers*. Generally speaking, the length of the SLRU chain cannot exceed VPSEQTxVPSIZE. Thus, VPSEQT limits the number of buffers that can be used by prefetch operations. The length of the SLRU chain can be determined from the SEQUENTIAL statistic. You can use the SEQUENTIAL statistic to predict whether modifying VPSEQT will have any effect. For example, if VPSIZE=50000 and VPSEQT=80, and SEQUENTIAL is hovering around 20000, lowering VPSEQT will not have any effect unless you lower VPSEQT below 40%.

A nonzero value for the VPSEQT HIT statistic means that VPSEQT was constraining the buffer pool consumption caused by sequential activity. RECLASSIFY is more difficult to interpret.

The VPSEQT and RECLASSIFY are statistics that are meant to be used by the IBM service representative.

The statistics are also reported in the IFCID 2 statistics record with one minor difference: Both the minimum (QBSTSMIN) and maximum (QBSTSMAX) value of SEQUENTIAL during the statistics interval are reported. IBM OMEGAMON® Performance Expert reports these numbers.

2.7 Change Data Capture Log read (IFCID 306) improvement

DB2 11 provides two major enhancements to DB2 log read processing. Both enhancements required a change in data replication products to support the new behavior.

- Continuous operations: Compression dictionary enhancement

  The DB2 instrumentation facility interface (IFI) provides the ability to read log records for data replication products to process the changes to a table from SQL update operation such as insert, update, and delete. If the source table is compressed, IFI needs a dictionary to decompress the log records of compressed data. When DB2 rebuilds a new compression dictionary, a problem can occur if a data replication product needs to read log records created using the old dictionary. The old compression dictionary is kept in memory while DB2 is up and running and is discarded after 24 hours or at shutdown of the DB2 system. In a data sharing group, the old compression dictionary is kept only in the memory of the member on which you run the utility (LOAD or REORG) to rebuild the dictionary, but the other members cannot access the dictionary in memory.

  DB2 11 NFM provides relief to this situation by saving the old compression dictionary to the log record and adding a record to the SYSIBM.SYSCOPY table to point to the RBA of the data sharing member's log record at which the compression dictionary is externalized. The change applies only to tables that are created with CHANGE DATA CAPTURE and COMPRESS YES.

- Performance: IFI filtering support

  Before DB2 11 filtering support, DB2 IFI READS of IFCID 306 return the log records of the tables defined as CHANGE DATA CAPTURE regardless of the capture tasks from the data replication product interested in the objects. With IFI filtering support added in DB2 11, capture tasks can qualify a list of DBID/PSID pairs to provide to DB2 IFI. Then, the DB2 IFI process uses the list to filter the log records and returns only the log records associated with the qualified objects. Since DB2 is now applying the filtering instead of the capture task, this can be done more efficiently reducing the cost of IFI and log read when only a subset of tables is qualified by capture tasks. Further benefit is possible if the source tables are compressed as IFI filtering can avoid the unnecessary decompression process for the log records that are filtered out.

  This enhancement is available in CM and is retrofitted to DB2 10 by APAR PM90568 (PTF UK97013).

  See also APAR II11809, an informational APAR for InfoSphere Replication Server for z/OS.

2.7.1 IBM InfoSphere Data Replication for DB2 for z/OS

To enable the enhancements, the user of IBM InfoSphere Data Replication (IIDR) for DB2 for z/OS requires version 10.2.1. The new level supports:

- Extended RBA/LRSN
- Compression dictionary enhancement
- IFI filtering

For details, start at the following site:

http://www-01.ibm.com/common/ssi/cgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS213-399

2.7.2 Performance improvement from IFI filtering

The degree of performance improvement from IFI filtering functions depends on the replication setup such as whether filtering can be applied, and the source tables are compressed or not. The following experiment was done by the IBM Montpellier benchmark team using InfoSphere Data Replication for DB2 for z/OS Q replication and DB2 for z/OS. The workload that is used is the IBM Financial Transaction Manager, which uses WebSphere Message Broker and WebSphere Transformation Extender, IBM ILOG®, DB2, and QREP, processing payment messages in batch mode. The connection between the WebSphere Message Broker and DB2 is ODBC/CLI. DB2 is configured as a four-member data sharing group across four separate LPARs. On each DB2 subsystem, a capture task is defined to manage its own set of tables updated by the member. A total of 20 tables are defined as CHANGE DATA CAPTURE enabled and each member processes five tables.

Measurement results

The comparison is done with DB2 10 and Q replication V10.1 against DB2 11 CM and Q replication V10.2.1. The source tables are not compressed in this test. The comparison shows:

- Overall workload throughput increased from 3242 tps to 3382 tps (+4.3%)
- Capture task, ASNQC101 class 2 CPU total is reduced from 431 seconds to 185 seconds.
- IFI filtering contributes CPU reduction as filtering is now done by DB2 IFI processing instead of the replication task.
- Apply task, ASNQA101 class 2 CPU stays similar between the two runs.

Conclusions on IFI filtering improvement

As mentioned, the improvement is highly dependent on the configuration, especially the number of tables that the replication tasks are interested in as opposed to the number of tables defined as CHANGE DATA CAPTURE. If compressed tables are filtered out, improvement can be significant. In the various IBM testing with Q replication or CDC, our observation on CPU reduction from capture tasks varies in the range of 15 - 60%. We have also observed CPU time reduction from the master address space. To realize this improvement, both DB2 (DB2 10 with PM90568 or DB2 11) and enhancement from data replication products to support filtering are necessary.

2.8 ACCESS DB command performance improvement

Sometimes the DBA would like to pre-open all of the data sets before the start of a workload period, such as the opening of the stock market in order to avoid the heavy overhead that would be associated with the start of the application. As a result, the transaction performance for the first SQL thread to reference a given page set or partition is improved.

Before DB2 9, the only way for pre-opening data sets was to execute simple dummy SQL that references those data sets. DB2 9 introduced the **`-ACCESS DATABASE MODE(OPEN)`** command to simplify the process of opening all of the data sets in a database, and to avoid the extra SQL overhead.

The DB2 command **-ACCESS DATABASE MODE(OPEN)** forces a physical open of a table space, index space, or partition on the local member. This moves the overhead of the physical open from an SQL thread to the command thread. As a result, the transaction performance for the first SQL thread to reference a given page set or partition is improved.

Because in DB2 9 and 10 the command was executed in serialized fashion, one task opening all data sets, causing long running time, many customers continued to use SQL because the SQL could be executed in parallel.

DB2 11 addresses this issue by using up to *20 threads* to execute the **-ACCESS DATABASE** command in parallel. Thus, the elapsed time of executing this command is significantly improved. This function is effective in DB2 11 CM.

The brokerage OLTP workload is used to demonstrate the performance benefit of this enhancement. The user database consists of 17,661 data sets. DB2 command, '**-ACC DB(user_db) SPACE(*) MODE(OPEN)**' is issued immediately after DB2 start.

Recording of elapsed time ends when the DSN9022I message appears. This message indicates that all synchronous processing for the specified command (**-ACC DB(user_db) SPACE(*) MODE(OPEN)**) completed successfully. Any tasks executing asynchronously for the specified command might still be executing when this message is displayed.

We observed a 761-seconds reduction in elapsed time, out of 980 seconds, with merely 3 seconds more CPU time. Overall, this enhancement provides parallelism support for the **-ACCESS DB** command resulting in 78% shorter elapsed time and minor CPU overhead with 20-way parallelism. The parallelism enhancement is retrofitted to DB2 10 via APAR PM80779.

Measurement data is highlighted in Table 2-4.

Table 2-4 Measurement data of -ACC DB MODE(OPEN) performance enhancement

	DB2 10	**DB2 11**	**Parallel/serial**
	Serial execution	**Parallel execution**	
Elapsed time (sec.)			
From '-ACC DB MODE(OPEN)' to DSN9022I message	980.13	219.81	77.6%
DBM1 (sec.)			
TCB	34.025	37.049	8.9%
Pre-emptible SRB	3.727	3.777	
Non-pre-emptible SRB	0.408	0.437	
Total	38.161	41.263	8.1%

z/OS optimization, SYSTEM MEMDSENQMGMT(ENABLE) in SYS1.PARMLIB(ALLOCxx), is applied to both measurements tabulated above. Enabling the MEMDSENQMGMT feature, available since z/OS 1.12, allows jobs and subsystems to use memory-based data set ENQ management for dynamically allocated data sets, which is faster than scheduler work area-based (SWA-based) data set ENQ management. When you enable this function on your z/OS system, data set ENQs for dynamically allocated data sets are managed in memory. We make a note here because the default setting is DISABLE. APARs PM17542 and PM46045 allow DB2 V8 (and DB2 9 and DB2 10) to take advantage of this z/OS optimization.

2.9 Security

DB2 11 adds important new functions that remove some usability constraints. Performance is not impacted by the new functions.

This section contains the following topics:

- Enhancements for exit authorization checking
- Enhancements for program authorization
- Removal of column access control restrictions
- AT-TLS encryption enhancement in DB2 10

2.9.1 Enhancements for exit authorization checking

DB2 11 for z/OS, when using DSNX@XAC exit authorization to obtain external security validation, has been enhanced to support:

- OWNER keyword for BIND and REBIND commands
- Owner authorization during autobind
- DYNAMICRULES(BIND) behavior for dynamic SQL statements
- DB2 cache entry refresh when RACF permissions change

These enhancements address some of the functional issues present in DB2 10, and they do not pose any significant performance impact. The overhead of initial authorization checking is normally negligible, and it occurs only once. Then, all subsequent authorization checking for the same packages, routines, and dynamic statements are done against caches in DB2, which is even more efficient and low cost.

> **Tip:** For more information about DB2 11 and exit authorization checking, see Chapter 10, "Security" of the Redbooks publication *DB2 11 for z/OS Technical Overview*, SG24-8180.

2.9.2 Enhancements for program authorization

DB2 11 provides authorization checking to verify if a program is a valid program that can execute a plan. This is done through a PROGAUTH BIND and REBIND PLAN option. Once enabled, a program and plan combination is inserted into user-defined table SYSIBM.DSNPROGAUTH, and the program authorization is checked against this table. Since this table has a unique index defined on PROGNAME and PLANNAME columns, there should not be any concern about performance.

2.9.3 Removal of column access control restrictions

In DB2 10 for z/OS, if you access masked columns through a query with GROUP BY or any of the set operations such as UNION, INTERCEPT, EXCEPT, error SQL code -20478 is issued. This has limited the usability of column masks for some users. In DB2 11, these restrictions have been removed. Many users who used to rely on views to restrict the access to or to limit what is shown in the columns due to the DB2 10 limitations can now consider dropping the views and migrating to column masks, which provide much tighter security and easier manageability.

In *DB2 10 for z/OS Performance Topics*, SG24-7942, we presented the performance comparison of controlling the access to columns using column masks versus equivalent views. The performance characteristics have not changed due to the new functional enhancements. Runtime performance of queries with column masks continues to be roughly the same as that of equivalent views.

2.9.4 AT-TLS encryption enhancement in DB2 10

In z/OS V1R12 Communication Server, the Application Transparent - Transport Layer Security (AT-TLS) processing has provided reduced CPU usage by avoiding unnecessary copying of data in and out of data space. This enhancement has significantly increased the throughput of encrypting and decrypting secured data within the z/OS environment. Even though it is not specific to DB2 11 for z/OS, we measured with DB2 10 and documented the results in this section, as a performance reference for your AT-TLS implementation.

> **Tip:** For configuring AT-TLS with DB2 for z/OS, refer to *DB2 10 for z/OS: Configuring SSL for Secure Client-Server Communications*, REDP-4799.

For AT-TLS policy rules, we used the parameters that are shown in Example 2-3.

Example 2-3 AT-TLS policy rule to support SSL Connection

```
TTLSEnvironmentAdvancedParms
{
   SSLv2 off
   SSLv3 off
   TLSv1 on
   TLSv1.1 off
      }
        TLSCipherParms
{
V3CipherSuites TLS_DHE_RSA_WITH_AES_256_CBC_SHA
}
```

To highlight the effect of this performance enhancement, we performed a before-and-after comparison using DB2 10 for z/OS running on z/OS V1.10 and z/OS V1.12. We turned off SSL Version 2, Version 3, and TLS Version 1.1 protocol so we can be sure that only TLS Version 1.0 protocol was used because it is supported by both z/OS V1.10 and V1.12. We also used 256-bit high strength encryption to stress the encryption processing.

We could not run DB2 11 for z/OS for this specific test because that would require z/OS V1.13 and we would not be able to do an apples-to-apples comparison test without the enhancement with DB2 11 for z/OS.

Figure 2-17 on page 50 shows the performance results of normalized throughput with an interactive workload that has light inbound and outbound network traffic. The higher the throughput is shown in the figure, the better the performance. After enabling 256-bit AT-TLS encryption, the normalized throughput dropped 15.5%. However, that percentage was reduced to just 8.4% after the enhancement that was made available in z/OS 1.12.

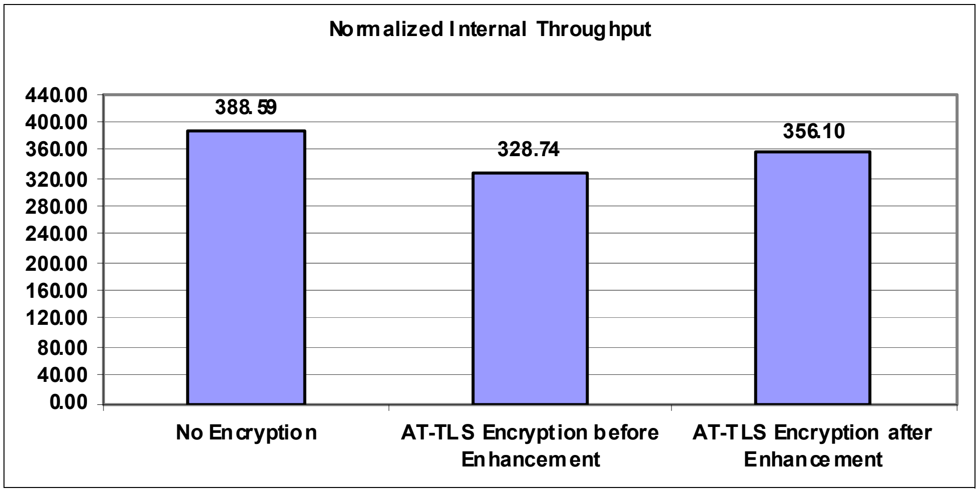

Figure 2-17 AT-TLS encryption performance enhancement with lightweight traffic workload

We then further stressed the encryption and decryption processing by using a network intensive workload transferring a large amount of encrypted data between a network client and DB2 for z/OS, just to highlight the difference. Figure 2-18 shows a 31.5% degradation after enabling 256-bit AT-TLS encryption with DB2 10 on z/OS 1.10, while with DB2 10 on z/OS 1.12 the degradation was reduced to 20%.

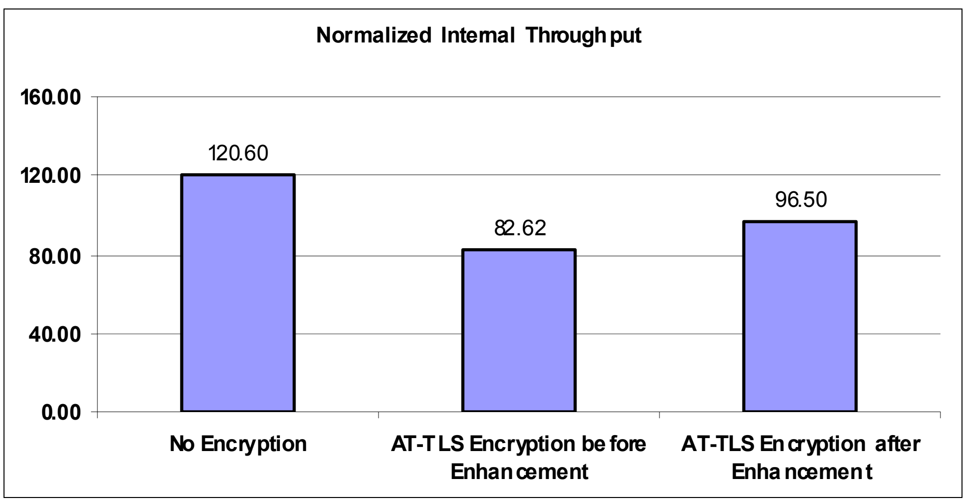

Figure 2-18 AT-TLS encryption performance enhancement with heavy network traffic workload

2.10 Scalability improvements with many partitions

DB2 Version 8 increased the maximum number of partitions for a table space from 256 to 4096 and users quickly began to define more than 256 partitions. Although a typical OLTP transaction accesses only a few partitions under each commit scope, it happens that, in DB2 10, as the number of defined partitions becomes large, the CPU time increases when RELEASE(COMMIT) is used. Thus, performance does not scale well as the number of defined partitions grows. Both universal table spaces (UTSs) and non-UTS are affected.

2.10.1 DB2 11 enhancement

DB2 11 resolves this scalability issue. Rather than the performance being sensitive to the number of partitions defined, the objective of DB2 11 is to only be sensitive to the number of partitions referenced under an individual COMMIT scope.

The performance improvement begins to be noticeable when there are more than 200 defined partitions in a table space. The more partitions there are, the larger the performance improvement. The biggest performance improvement is found for applications issuing a single SELECT statement that touches only one partition out of 4096.

This improvement is effective in DB2 11 CM and no user action is necessary. No REBIND needed. This enhancement is effective with packages bound with the RELEASE(COMMIT) option; RELEASE(DEALLOCATE) applications are not affected.

2.10.2 Performance measurements

The following performance measurements were conducted to demonstrate the CPU reduction from accessing a few partitions, out of many defined partitions, within a COMMIT scope in DB2 11.

Figure 2-19 shows that DB2 10 exhibits a non-linear class 2 CPU time increase as the number of defined partitions increases, while DB2 11 exhibits a consistent performance. It affirms that the performance of an application bound with RELEASE(COMMIT), is sensitive to the number of partitions accessed within a COMMIT scope rather than to the number of defined partitions. Figure 2-19 shows the performance of classic partitioned table space.

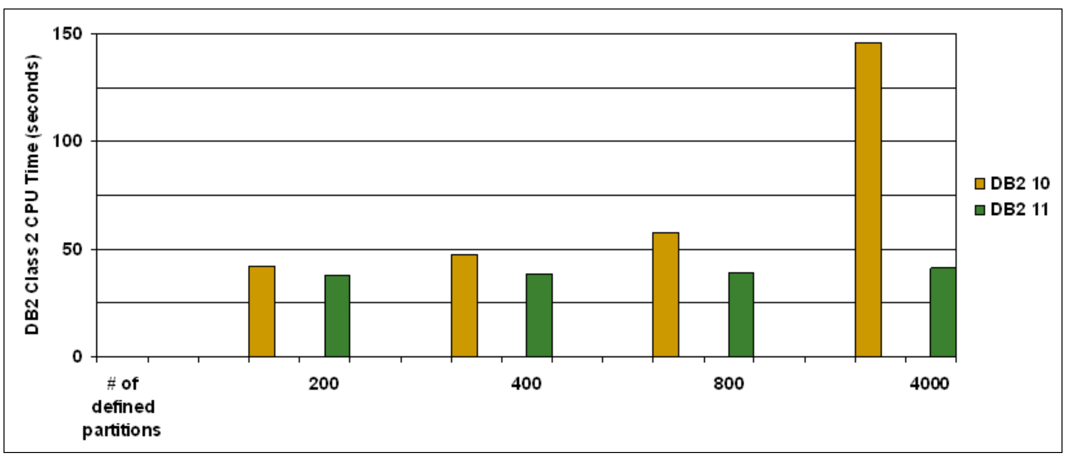

Figure 2-19 Select from one partition within a commit scope: Classic partition table space

Figure 2-20 shows the performance for UTS partition by range. UTS and non-UTS show a similar trend.

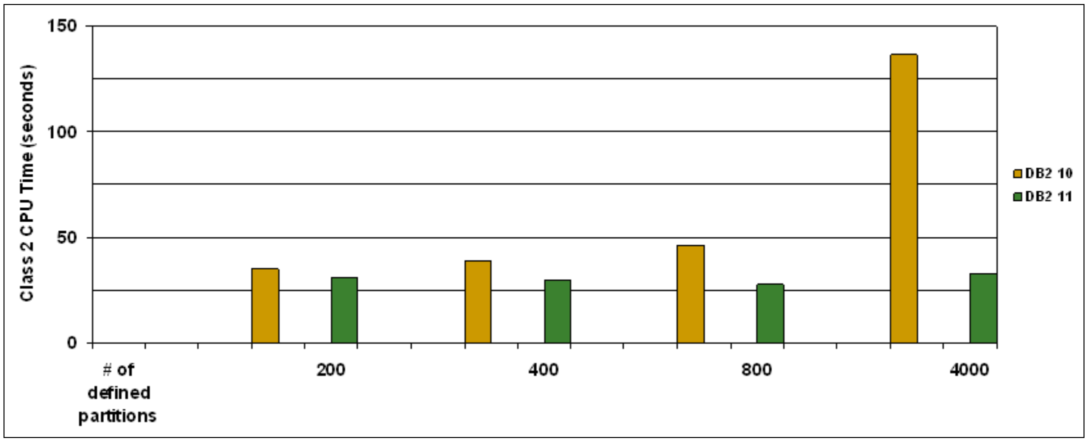

Figure 2-20 Select from one partition within a commit scope: UTS, partition by range

Figure 2-21 shows UTS partition by growth. MAXPARTITIONS in CREATE TABLESPACE DDL defines the maximum number of partitions for UTS partitioned by growth. Again, we see that DB2 10 shows a non-linear CPU time increase as the MAXPARTITIONS increased, and that DB2 11 exhibits consistent performance as the number of partitions defined increases.

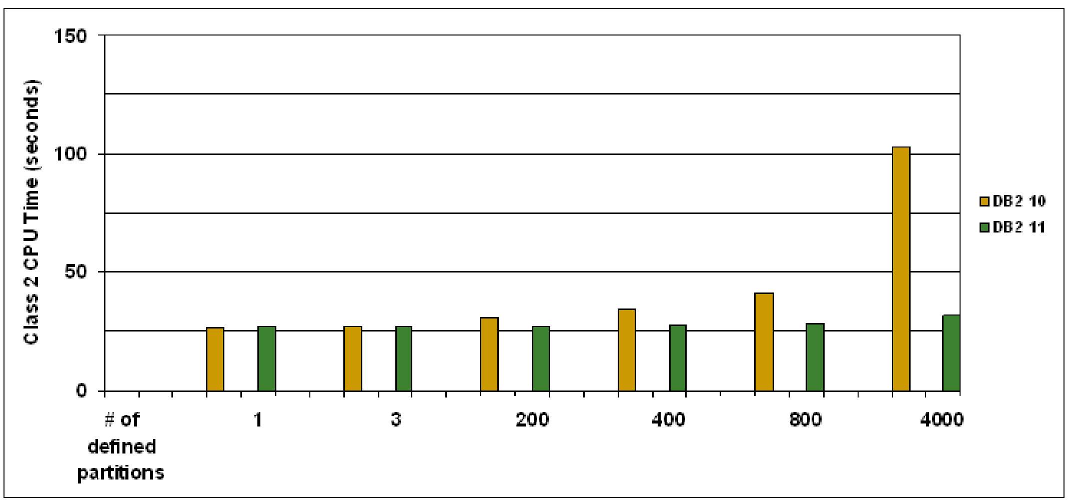

Figure 2-21 Select from one partition within a commit scope: UTS, partition by growth

With this DB2 11 optimization, the performance of accessing the same partitions within a COMMIT scope remains constant with an increasing large number of partitions defined. This is shown for classic partition table space, UTS partition by range, and UTS partition by growth.

The performance improvement of a Brokerage OLTP workload due to this optimization is shown in Figure 2-22 on page 53. There are 12 out of 32 UTS partition by range table spaces having 1000 partitions defined. All applications, which are selecting and updating data, are bound with RELEASE(COMMIT). The measurement results show 2.4% CPU time improvement. The SAP Posting workload also benefits from this enhancement, see 2.14, “SAP Banking Service Day Posting workload measurements” on page 60.

In general, with few partitions defined (less than 200) or if all partitions are accessed, there is no difference. RELEASE (DEALLOCATE) was not impacted by the number of partitions and is not changed by this enhancement.

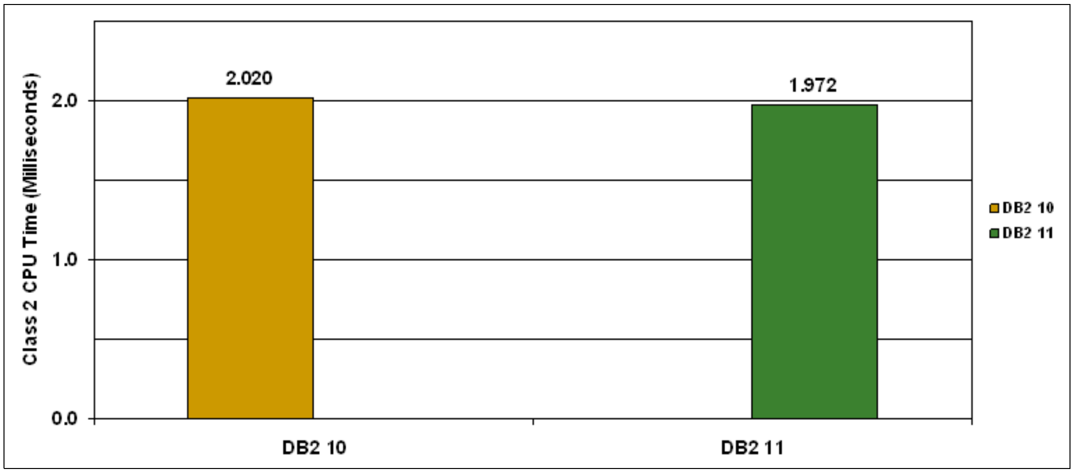

Figure 2-22 Brokerage OLTP workload and large number of partitions with RELEASE(COMMIT)

Applications that are similar to the SAP Banking Service Day Posting with many partitions will benefit from this enhancement.

2.11 Persistent thread break-in

With the removal of the virtual storage constraints, the BIND option RELEASE (DEALLOCATE) has been made again available for DRDA packages and can be considered to improve the performance of frequent critical transaction for local and distributed environments.

DB2 10 introduced a method that allows to switch between RELEASE(COMMIT) and RELEASE(DEALLOCATE) BIND options for existing distributed applications without having to rebind, so that activities such as running bind, DDL, and utilities, can happen without an outage.

DB2 10 honors the RELEASE bind option (which causes a persistent thread) for database access threads. This minimizes the use of CPU resources for package allocation and deallocation processing, resulting in performance improvements. You can modify this behavior by using the **`MODIFY DDF PKGREL`** command for distributed packages. In releases before DB2 10, the RELEASE bind option had no effect on distributed database access threads, the option COMMIT was implicitly applied as of Version 6.

See Chapter 9, “Connectivity and administration routines” of the Redbooks publication *DB2 10 for z/OS Technical Overview*, SG24-7892.

2.11.1 Break into persistent threads

DB2 11 introduces the ability for the Rebind, DDL, and online REORG to break into persistent threads on a transaction boundary for active and for idle threads bound with RELEASE(DEALLOCATE), distributed or non-distributed.

CICS, IMS, and RRS typically set aside agents as part of their thread pooling process. These agents can remain idle for a long duration holding package locks. Before this enhancement, until these threads become active, a requester for an X-lock on a given package (for example, a job trying to Rebind that package) would timeout.

Based on the IRLM timeout interval, a broadcast message is now sent to all DB2s to recycle idle threads holding package locks for which there are requesters for X-lock. Recycling causes the resources held by the idle threads to be released as though it was bound with RELEASE(COMMIT) instead of RELEASE(DEALLOCATE). The idle thread should have issued an explicit commit for it to be eligible for recycling.

The new break-in support can also be a replacement for the **MODIFY DDF** command: Users do not need to switch BIND options for distributed transactions.

2.11.2 Performance implications

When Rebind package is able to break-in while active transactions using that package are running, then all subsequent threads using that package are queued until the completion of the rebind. This can cause elongated response times for those transactions. The increase in response time is due to an increase in lock suspension times.

Important: Depending on your environment and the frequency of changes, you might want to consider increasing the DDL lock timeout value.

Figure 2-23 shows one example where Rebind packages were issued during the execution of the IRWW OLTP workload on a two-way data sharing system. The packages were bound with RELEASE(DEALLOCATE). Note the temporary but significant increase in the Lock/Latch suspension times for the transactions whose packages were being rebound.

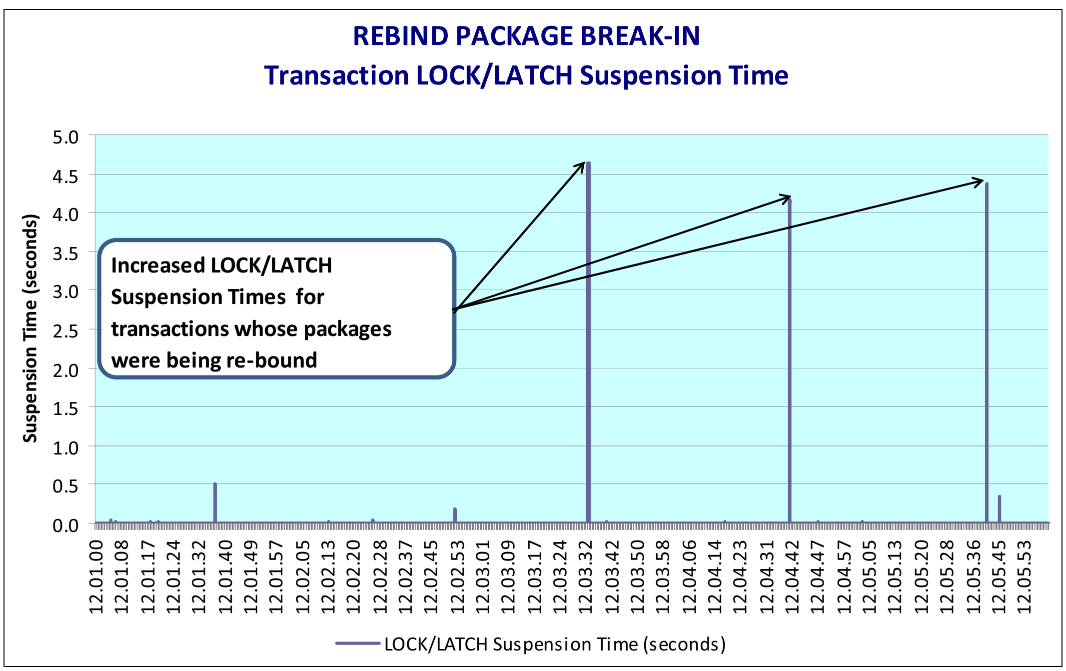

Figure 2-23 REBIND package break-in

2.12 Classic IRWW workload measurements

In this section, we describe the results of some key performance measurements using the Classic IRWW workload. The transactions use local IMS attach to connect to DB2 for z/OS.

See Appendix B.1, “IBM Relational Warehouse Workload” on page 298.

2.12.1 DB2 10 to DB2 11 migration scenarios

We conducted a series of performance tests in a sequence that mimics the order in which some customers might upgrade their DB2 for z/OS subsystems from DB2 10 to DB2 11.

These measurements were conducted in a one-way data sharing setup (that is, the DB2 environment was set up for data sharing, but only one member in the data sharing group was active). The LPAR used was on z196 with two dedicated CPs and 48 GB of main storage. Table 2-5 summarizes the measurements from these tests.

Table 2-5 Measurement for Classic IRWW - RELEASE(COMMIT) - data compressed - z/OS Version 1 Release 13

Measurement for Classic IRWW	ITR (commit/sec)	Total CPU/commit in microseconds	DB2 10 to DB2 11 CPU% change
Baseline - DB2 10 NFM	2,022.5	627	
DB2 11 CM no REBIND	2,020.9	627	0.0
DB2 11 CM with REBIND	2,004.9	630	0.5
DB2 11 NFM with REBIND	2,029.3	622	-0.8
DB2 11 with extended LRSN	2,102.0	587	-6.4
DB2 11 with extended LRSN and z/OS 2.1	2,112.8	578	-7.8

Internal throughput rate (ITR) is the COMMITs/sec normalized to 100% CPU busy. The first four measurement results are within the measurement noise since the IRWW transaction CPU usage is shown in micro seconds.

The data demonstrates significant CPU reduction for this workload as a result of converting the DB2 objects (table spaces and index spaces) to use extended RBA/LRSN. Not all transactions benefitted equally from the extended RBA/LRSN. The transactions that perform multiple updates and deletes on the same page benefitted the most.

2.12.2 DB2 11: zEC12 versus z196

We compared the performance of Classic IRWW workload on z196 and zEC12. These measurements were conducted on z196 and zEC12 using 2 dedicated CPs and 48 GB of main storage. The DB2 11 environment was configured as one-way data sharing.

The measurements were done with DB2 11 NFM and packages that are bound with RELEASE(COMMIT).

Measurements were done with one-way data sharing since we do not maintain a non-data sharing system and the performance should be equivalent.

Table 2-6 shows the observations from those measurements.

Table 2-6 Classic IRWW Measurements: zEC12 versus z196

Metric	z196 (A)	zEC12 (B)	% Change ((B-A)/A)
ITR - Commits/sec	2,029	2,679	32
Total Service Units/Commit	36.32	34.42	-5.2

The data shows that for this workload, both the throughput and Service Units improved significantly on zEC12 when compared to the usage on z196.

2.13 High insert batch workload measurements

In this section, we examine the performance of intensive batch concurrent inserts using the following scenarios:

- Random insert and delete
- Journal table inserts
- Message queue inserts and deletes
- APPEND and MEMBER CLUSTER with sequential key insert

The main performance factor is the cost of searching for space, which depends on the type of table space. Given a very large table space, the best way to improve insert concurrency is to divide the rows into partitions and to define a partitioning index to be the cluster index. This prevents inserts into different partitions from latching or updating the same space map pages.

Other factors contribute to the performance effects, such as the COMMIT frequency, LOCKSIZE, free space, MEMBER CLUSTER, MAXROWS, SEGSIZE, APPEND, and page size. For practical reasons, we cannot examine all combinations of these parameters, and can only focus on measurements of a few of them. All of the measurements were performed using a zEC12 processor with two LPARs.

The environment consists of the following components:

- 100 partitions table with page level locking
- 300 threads
- DB2 11 two-way data sharing members

2.13.1 Random insert and delete

We start by creating a 100 partitions using page level locking. We initially seeded 40 million rows into the table. Then, we measured the random insertion of 4 million rows while different threads deleted 2.7 million rows. The threads were spread across the two members.

Both classic partitioned table spaces (PTS) and partition by range (PBR) were tested. Figure 2-24 on page 57 shows the throughput results in terms of the total number of SQL statements per second (one row was inserted or deleted per statement.)

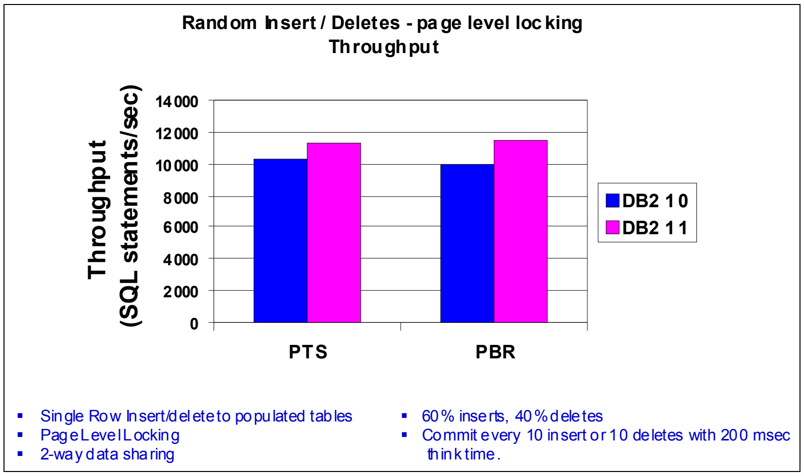

Figure 2-24 Random Insert/Deletes - page level locking - Throughput

Figure 2-25 shows the average class 2 CPU time for these SQL statements.

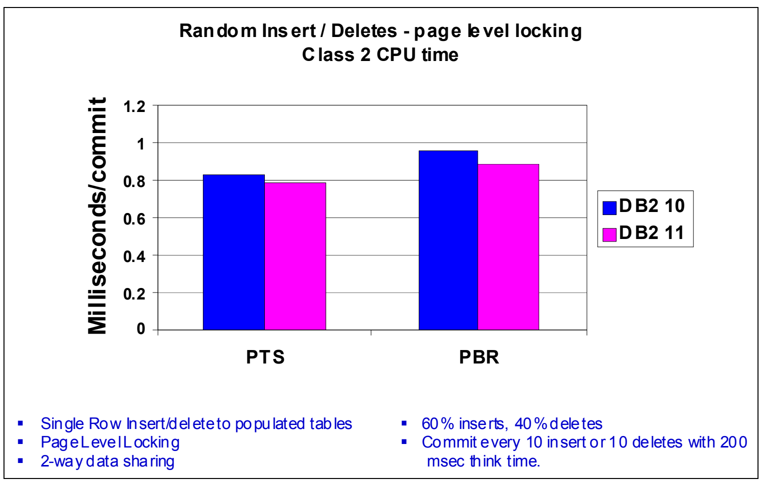

Figure 2-25 Random Insert/Deletes - page level locking - class 2 CPU Time

Approximately 60% of the SQL statements were inserts while 40% of the SQL statements were deletes, but the percentages were not identical between DB2 10 and 11 due to some randomness. A commit was done after every 10 inserts or 10 deletes, with a 200 millisecond think time after each SQL statement.

The results showed that DB2 11 increased the throughput of PTS table spaces by 8.3% and increased the throughput of PBR table spaces by 14.5%. With DB2 11, the throughput of PBR was 1.7% higher than PTS. DB2 11 decreased the class 2 CPU time of PTS by 5.5% and that of PBR by 7.2%. However, with DB2 11, PBR used 12.5% more class 2 CPU time than PTS.

2.13.2 Journal table inserts

The next test simulated what we call a *journal table*, that is, sequential inserts. However, these inserts were done with and without MEMBER CLUSTER (MC). Only PBR was used for these tests and row level locking was used. This time, we used multi-row inserts of 100 rows per INSERT.

Figure 2-26 shows the throughput.

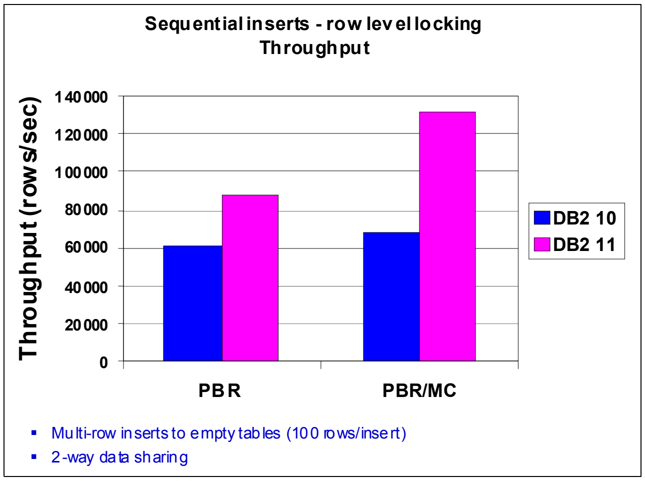

Figure 2-26 Sequential inserts - row level locking - Throughput

DB2 11 increased the throughput without member cluster by 45% and it increased the throughput of member cluster by 95%. With DB2 11, the throughput of member cluster was 51% higher than without member cluster.

Figure 2-27 shows class 2 CPU time of the inserts.

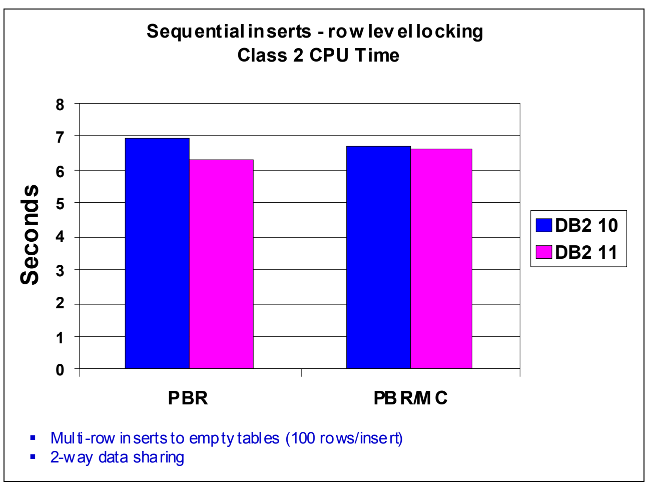

Figure 2-27 Sequential inserts - row level locking - class 2 CPU Time

DB2 11 reduced the class 2 CPU time without member cluster by 9.2% and had no effect on the CPU time with member cluster.

2.13.3 Message queue inserts and deletes

The next test simulated a message queue management application. The table was treated like a queue. A task inserted some rows sequentially to this queue while it began to delete the rows at the front of this queue. Ten rows were inserted and then 10 rows were deleted. It is a multi-row insert with 10 rows inserted per commit.

The queue started in an empty state and finished in an empty state. A partition by growth (PBG) table space was used with row level locking.

Figure 2-28 shows the throughput in terms of rows/second.

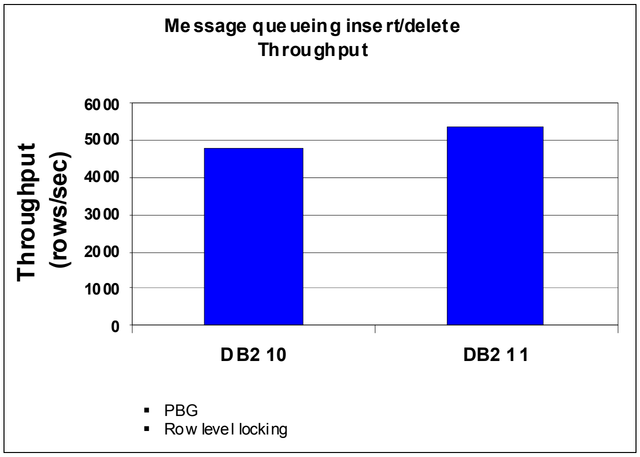

Figure 2-28 Message queuing insert/delete: Throughput

Figure 2-29 shows the class 2 CPU time in terms of milliseconds per commit.

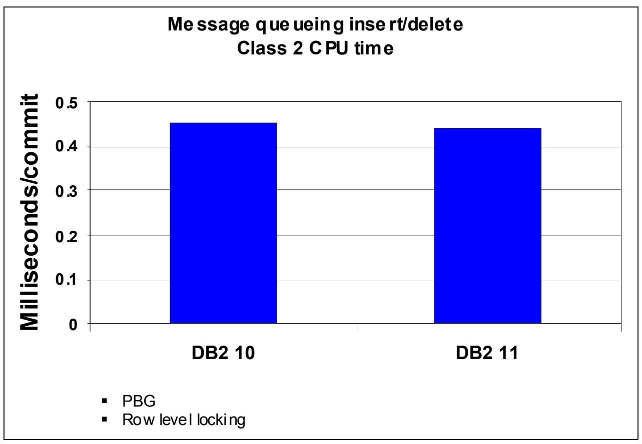

Figure 2-29 Message queuing insert/delete: class 2 CPU Time

DB2 11 increased the throughput by 11.6%. CPU time was unchanged.

2.13.4 APPEND and MEMBER CLUSTER with sequential key insert

The next tests were done to show how, with DB2 11, member cluster affects the performance sequential key inserts when the APPEND option is used with row level locking. It is a multi-row insert. One hundred rows were inserted into three tables, then commit was issued.

Figure 2-30 shows that member cluster increased the throughput by 10.7%.

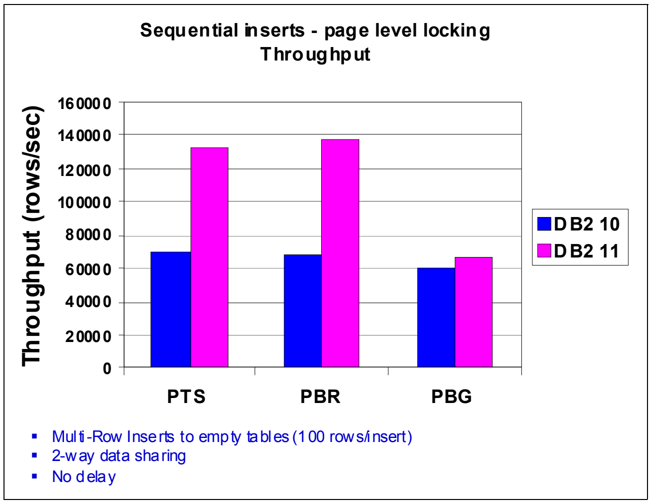

Figure 2-30 Sequential inserts - page level locking - Throughput

Figure 2-31 shows that it reduced the class 2 CPU time by 13%.

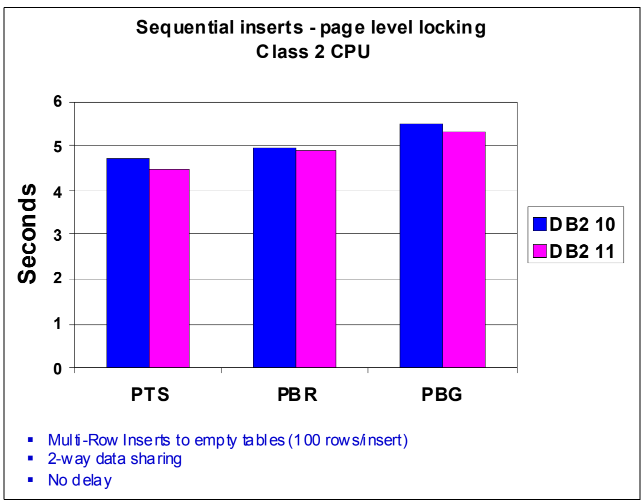

Figure 2-31 Sequential inserts - page level locking - class 2 CPU Time

2.14 SAP Banking Service Day Posting workload measurements

In this section, we describe the summary of measurements conducted by the IBM SAP on System z Performance Team of Systems and Technology Group. The measurements are done to demonstrate the performance expectation of DB2 11 with SAP online transaction processing workload. Detailed information can be found in the IBM zEnterprise System: DB2 11 for z/OS with SAP Performance Report published by the IBM SAP Performance Team at the following site:

http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP102394

2.14.1 Workload description

The SAP Banking Services Day Posting workload was developed by SAP to simulate customer environments. The workload consists of interactive users going through repetitive cycles of 15 dialog steps creating postings and performing other account operations. A posting is a deposit or a withdrawal from a customer's account. Typical examples of a posting are a payment out of the account or a deposit into the account.

2.14.2 Measurement environment

All test scenarios were done using a physical three-tier environment where each of the three layers resided on separate machines. The SAP Database Server was on an IBM zEnterprise System (zEC12) running z/OS. The SAP application servers were IBM p7 Blade Servers running IBM AIX®. The presentation server was an IBM POWER5 p55A running AIX.

2.14.3 Measurement results

The focus was on areas that are important to customers running SAP on System z. Various configurations were used to show how DB2 11 performed in different environments. Comparable tests were executed with DB2 10.

DB2 single system: 5 M accounts (SBS 6.0)

In these measurements, the SAP Day Posting workload was run with the SBS 6.0 database with 5 M accounts. DB2 10 and DB2 11 CM were run on a zEC12 with a single z/OS LPAR with 2 CPs. There was a 7% improvement in ITR with DB2 11. The SBS 6.0 5 M account database had the key banking tables defined with 128 partitions.

DB2 single system: 60 M accounts (SBS 7.0)

In these measurements, the SAP Day Posting workload was run with the SBS 7.0 database with 60 M accounts. DB2 11 was in NFM using the large 10-byte RBA format. A series of measurements were run on a zEC12 with a single z/OS LPAR with 2 CPs, 4 CPs, and 8 CPs.

The results that are shown in Figure 2-32 on page 62 were impressive. A 15% improvement in ITR with DB2 11 on the zEC12 with 2 CPs. There was a 16% improvement in ITR with 4 CPs. And there was an 18% improvement in ITR with 8 CPs. These results show that DB2 11 also delivers efficient CP scaling. As the DB2 system got larger, the performance improvement of DB2 11 compared to DB2 10 grew.

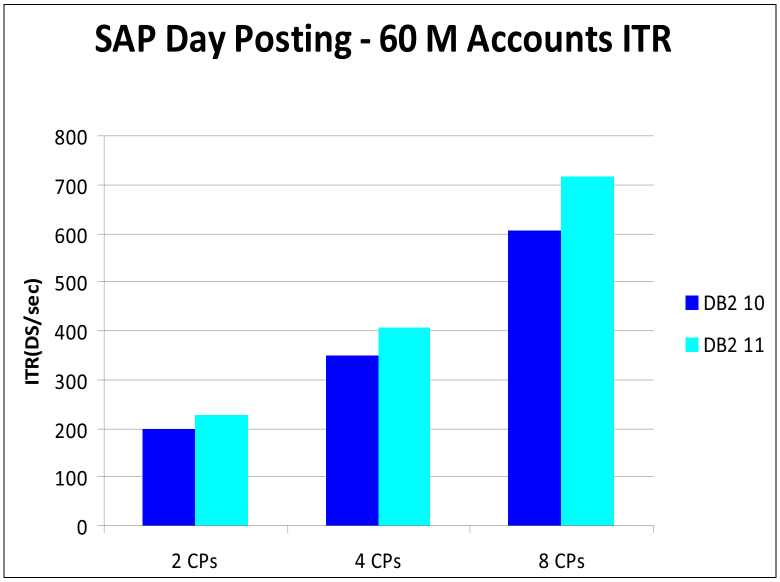

Figure 2-32 DB2 10 and 11: Throughput with 60 M accounts in SAP Day Posting

A significant contributor to the performance improvements seen with DB2 11 in these measurements was the performance improvement, which relates to the large number of partitions in a database. See 2.10, “Scalability improvements with many partitions” on page 50. The SBS 7.0 60 M account database had the key banking tables defined with 512 partitions. Systems with databases with a similar number of partitions to this may see performance improvements in this range.

Another significant contributor to the observed performance are the enhancements made in DB2 11 in the area of column processing. Since a typical SAP application has large numbers of columns in the select list, this optimization is effective especially in SAP applications. See 2.4, “Column processing runtime optimization” on page 38.

Additional contributors: Log buffer improvement and DDF Sync Receive mode change.

DB2 10 provided the significant relief in virtual storage usage below the 2 GB bar in DBM1. DB2 11 continues the trend to reduce the usage of below the bar storage by moving xProc storage above the bar. There was a 48% reduction in DBM1 below the bar virtual storage usage with DB2 11 compared to DB2 10 and equivalent real storage usage. There was no difference in virtual storage usage in the DIST address space between DB2 10 and 11.

These measurements with the SBS 7.0 60 M account database cannot be compared with the one done with the SBS 6.0 5M account database. The differences in the SAP levels (SBS 6.0 versus SBS 7.0), the database size (5 M accounts versus 60 M accounts), and the number of database partitions (128 versus 512) make it like different workloads were used, even though both sets of measurements used the SAP Day Posting workload.

DB2 two-way data sharing: 60 M Accounts (SBS 7.0)

In these measurements, the SAP Day Posting workload was run with the SBS 7.0 database with 60 M accounts in a DB2 two-way data sharing environment. DB2 10 and DB2 11 were run. DB2 11 was in NFM using the extended 10-byte RBA/LRSN. These measurements were run on a zEC12 with two z/OS LPARs, each with 6 CPs. There was a 9% improvement in ITR with DB2 11.

The performance benefit of the full LRSN spin avoidance delivered in DB2 11 in this data sharing measurement was not noticed due to the random access nature of the Day Posting workload. However, the benefits of this should be seen in batch workloads with more sequential access.

Similar trends in virtual storage and real storage usage in two-way data sharing were seen in a single system. There was a 31% reduction in virtual storage usage in DB2 11 compared to DB2 10. There was less than a 5% increase in real storage usage with DB2 11.

3

Synergy with z platform

DB2 11 for z/OS takes advantage of the latest improvements in the z platform. DB2 11 increases the synergy with System z hardware and software to provide better performance, more resilience, and better function for an overall improved value. It also adds more System z Integrated Information Processor (zIIP) eligibility.

DB2 benefits from large real memory and faster processors as well as large page frame support. DB2 uses the enhanced z/OS features for catalog access, Flash Express, and virtual storage reduction. See 11.1.2, “DB2 related enhancements in z/OS 2.1” on page 249 for a summary of z/OS 2.1 enhancements of interest to DB2.

In this chapter, we discuss the following topics:

- Additional zIIP redirect with DB2 11
- Large frames
- RoCE network adapters
- 1 MB code pages
- 200,000 open data sets
- Query parallelism storage-related reduction
- Exploitation of large memory for buffer pools

3.1 Additional zIIP redirect with DB2 11

The IBM System z Integrated Information Processor (zIIP) was introduced in 2006. The zIIP is designed to help free up general computing capacity and lower software costs for selected subsystem workloads.

DB2 for z/OS began using zIIP specialty processors in Version 8 and has continued to improve the total cost of ownership (TCO) by further using zIIP engines in DB2 9 and DB2 10. DB2 11 continues this trend by providing additional zIIP workload eligibility, as we describe in this section.

The initial DB2 implementation of zIIP was targeted towards reducing the customer relationship management (CRM) and in general distributed workloads on the mainframe. In addition, portions of DB2 LOAD, REORG, REBUILD INDEX, RUNSTATS utilities processing, CPU intensive parallel query processing, and XML document schema parsing processing were made eligible for zIIP redirect. The amount of redirect in each case varies based on workload characteristics.

3.1.1 Redirect for the DB2 address spaces

The following is the additional DB2 11 for z/OS processing that is authorized to execute on a zIIP[1]:

Asynchronous processing activity that is executed under enclave service request blocks (SRBs) and is “charged” for CPU consumption purposes to an IBM DB2 address space (rather than to user applications), except for P-lock negotiation processing, is authorized to execute on a zIIP with DB2 11.

From the DB2 address spaces point of view, such zIIP eligible processing includes:

- DBM1 address space:
  - System task performing clean-up of pseudo deleted index entries.
  - Portions of XML multi-version documents cleanup processing (also available in DB2 10 via APAR PM72526).
  - System-related asynchronous SRB processing except for P-lock negotiation processing.
- MSTR address space:

  System-related asynchronous SRB processing, such as log write or log read

Refer to the IBM documentation for software and hardware requisites for zIIP at “IBM System z Integrated Information Processor (zIIP)”.

Example 3-1 on page 67 shows an extract from the OMPE Statistics report CPU section showing the zIIP redirect for the DB2 address spaces.

[1] This information provides only general descriptions of the types and portions of workloads that are eligible for execution on Specialty Engines (e.g, zIIPs, zAAPs, and IFLs) ("SEs"). IBM authorizes customers to use IBM SE only to execute the processing of Eligible Workloads of specific Programs expressly authorized by IBM as specified in the "Authorized Use Table for IBM Machines" provided at www.ibm.com/systems/support/machine_warranties/machine_code/aut.html ("AUT"). No other workload processing is authorized for execution on an SE. IBM offers SE at a lower price than General Processors/Central Processors because customers are authorized to use SEs only to process certain types and/or amounts of workloads as specified by IBM in the AUT.

Example 3-1   OMPE Statistics report: CPU section

CPU TIMES	TCB TIME	PREEMPT SRB	NONPREEMPT SRB	CP CPU TIME	**PREEMPT IIP SRB**	CP CPU /COMMIT
SYSTEM SERVICES ADDRESS SPACE	9.081149	0.082266	0.588854	9.752269	2.079948	0.000284
DATABASE SERVICES ADDRESS SPACE	0.789612	9.586555	0.088727	10.464893	18:35.367575	0.000304
IRLM	0.001099	0.000000	9.767394	9.768493	0.000000	0.000284
DDF ADDRESS SPACE	1.844648	36:46.086825	1.920420	36:49.851894	46:00.841539	0.064294
TOTAL	11.716507	36:55.755646	12.365395	37:19.837548	1:04:38.289062	0.065166

In this example, the “PREEMPT IIP SRB” column shows the actual zIIP redirect CPU in seconds for the Statistics interval. The “CP CPU/commit” value does not include the 'PREEMPT IIP SRB' CPU.

Before DB2 10, zIIP redirect was shown for the DDF (DIST) address space for the DRDA workload zIIP redirect and the DBM1 address space for a parallel query.

DB2 10 added the reporting of the zIIP redirect for the prefetch and deferred writes processing under DBM1 (Database Services Address Space). For applications that do high prefetch processing, the zIIP redirect can be a significant percentage of the DBM1 address space CPU.

DB2 11 redirects additional asynchronous SRB processing such as pseudo deleted index entries cleanup and XML multi-version documents cleanup in the DBM1 address space. zIIP redirect for the MSTR (System Services) address space asynchronous SRB processing such as log write and read, is new in DB2 11. zIIP redirect for the MSTR address space generally is less than the DBM1 address space, except for commit intensive applications. There is no zIIP redirect for the IRLM address space.

3.1.2 Additional zIIP redirect for the DB2 utilities

DB2 for z/OS began using zIIP specialty processors for Utilities in Version 8 and continued to improve total cost of ownership (TCO) by further increasing the zIIP eligibility workload in later DB2 versions.

The following DB2 11 for z/OS processes are authorized to execute on a zIIP:

- The DB2 base LOAD, REORG, and REBUILD INDEX utility processing of inline statistics collection that DB2 directs to be executed under enclave SRBs.[2]
- The DB2 processing of inline statistics and the base processing of RUNSTATS utility column group distribution statistics collection that DB2 directs to be executed under enclave SRBs.
- The DB2 base LOAD utility index management processing when running LOAD REPLACE that DB2 directs to be executed under enclave SRBs.

Refer to the IBM documentation for software and hardware requisites for zIIP at “IBM System z Integrated Information Processor (zIIP)”.

Our measurements on four-way zEC12 and one zIIP processor showed:

- 14% LOAD CPU is eligible for zIIP offload, 37% LOAD with inline statistic CPU is eligible for zIIP offload.
- 18% REORG CPU is eligible for zIIP offload, 41% REORG with inline statistic CPU is eligible for zIIP offload.
- Up to 18% REBUILD INDEX CPU is eligible for zIIP offload, 33% REBUILD INDEX CPU with in-line statistic CPU is eligible for zIIP offload.

[2] DB2 does not direct all such base utility processing to be executed under enclave SRBs.

Details of the results of the measurements are in Table 3-1. All times are in seconds.

Table 3-1 zIIP eligibility for utilities

Table of 20 partitions 100 M rows, 6 indexes	**CPU (CP)**	**zIIP CPU**	**zIIP CPU on CP**	**Total CPU**	**Elapsed time**	**% zIIP eligible**	**% off loaded**
LOAD TABLE	295.44	38.06	7.97	333.5	195.73	**14%**	11%
LOAD STATISTICS TABLE	309.40	153.91	15.75	463.31	238.70	**37%**	33%
LOAD STATISTICS TABLE COLGROUP	351.37	181.22	16.38	532.59	275.72	**37%**	34%
LOAD STATISTICS TABLE COLGROUP HISTOGRAM	352.50	188.02	15.96	540.52	283.12	**38%**	35%
REORG TABLE	268.41	55.49	4.31	323.90	258.40	**18%**	17%
REORG STATISTICS TABLE	328.04	122.18	60.63	450.22	306.77	**41%**	27%
REORG STATISTICS TABLE COLGROUP	368.76	161.61	52.67	530.37	345.61	**40%**	30%
REORG STATISTICS TABLE COLGROUP HISTOGRAM	363.46	168.20	52.83	531.66	351.72	**42%**	32%
REBUILD PI	37.27	6.40	1.21	43.67	16.67	**17%**	15%
REBUILD PI PART 1	1.64	0.36	0	2.00	3.65	**18%**	18%
REBUILD NPI	71.62	10.54	0.03	82.16	52.36	**13%**	13%
REBUILD NPI PART 1	3.52	2.84	0	6.36	16.43	**45%**	45%
REBUILD 6 INDEXES	283.26	34.26	9.27	317.52	103.39	**14%**	11%
REBUILD PI STATISTICS	44.96	10.97	6.98	55.93	21.14	**32%**	20%
REBUILD PI PART 1 STATISTICS	1.85	0.93	0	2.78	3.70	**33%**	33%
REBUILD NPI STATISTICS	73.83	24.24	0.03	98.07	53.46	**25%**	25%
REBUILD NPI PART 1 STATISTICS	3.53	2.87	0	6.40	16.43	**45%**	45%
REBUILD 6 INDEXES STATISTICS	330.43	55.03	41.49	385.46	114.65	**25%**	14%

Only portions of the utility processing run on enclave SRB. 100% of the enclave SRB processing is zIIP eligible. The zIIP eligibility column is shown in bold in Table 3-1. In our measurements, we were restricted by having only 1 zIIP engine, hence the smaller % offloaded. For sizing zIIP solutions, see the article “Performance and Capacity Considerations for zIIP Engines” at the following link:

http://ibmdatamag.com/2013/04/performance-and-capacity-considerations-for-ziip-engines

3.2 Large frames

This chapter describes the following two topics that are related to large z/OS memory frames:

- 2 GB frames
- Flash Express and pageable large pages

3.2.1 2 GB frames

In addition to basic 4 KB frames, z/OS supports large-size frames that can be 1 MB and, starting with z/OS 1.13 and EC12, 2 GB. Using 1 MB page frames saves CPU consumption by improving the hit ratio of the system translation lookaside buffer (TLB).

In DB2 10, 1 MB frames are non-pageable and are used with the DB2 buffer pool parameter PGFIX(YES). To use 1 MB or 2 GB frames, you must specify the LFAREA system parameter to specify either 1 MB frames or 2 GB frames, or a mixture of the two.

The IEASYSxx LFAREA parameter specifies the amount of real storage to be made available for 1 MB and 2 GB large pages. All 1 MB and 2 GB pages are backed by contiguous 4 KB real storage frames, and are allocated from real storage as specified by the LFAREA parameter. If the system becomes constrained by a lack of sufficient 4 KB pages to handle workload demand, it can use free 1 MB large pages to back 4 KB page requests, enabling the system to react dynamically to changing system storage frame requirements.

If additional 1 MB large pages are required, the system can recombine 4 KB frames that had been taken from the 1 MB large page area and place them back into the LFAREA as 1 MB pages. However, frequent use of this decomposition and recombination function can indicate a system configuration and tuning issue: The large-page allocation (LFAREA) might be too large, or the demand for 4 KB frames might be higher than expected. To resolve this issue, you can either decrease the size of the large page frame area (LFAREA) or adjust the workload to reduce the demand for 4 KB frames.

Example 3-2 shows the output of the **DISPLAY VIRTSTOR,LFAREA** MVS system command.

Example 3-2 DISPLAY VIRTSTOR,LFAREA MVS system command

```
D VIRTSTOR,LFAREA
IAR019I  13.41.09 DISPLAY VIRTSTOR
 SOURCE =  1D
 TOTAL LFAREA = 88064M , 0G
 LFAREA AVAILABLE = 87641M , 0G
 LFAREA ALLOCATED (1M) = 423M
 LFAREA ALLOCATED (4K) = 0M
 MAX LFAREA ALLOCATED (1M) = 68007M
 MAX LFAREA ALLOCATED (4K) = 0M
 LFAREA ALLOCATED (PAGEABLE1M) = 0M
 MAX LFAREA ALLOCATED (PAGEABLE1M) = 0M
 LFAREA ALLOCATED NUMBER OF 2G PAGES = 0

 MAX LFAREA ALLOCATED NUMBER OF 2G PAGES = 0
```

Use this command to verify the following LFAREA settings:

- Source of LFAREA parameter can be a parmlib member, operator-supplied, or the default.
- The size of both the 1 MB LFAREA, in megabytes, and 2 GB LFAREA, in gigabytes, in decimal.
- The amount of both the 1 MB LFAREA, in megabytes, and 2 GB LFAREA, in gigabytes, that is available, in decimal.

- The amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of 1 MB-page requests.
- The amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of 4 KB-page requests.
- The high water mark for the amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of 1 MB-page requests.
- The high water mark for the amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of 4 KB-page requests.
- The amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of pageable 1 MB-page requests.
- The high water mark for the amount of 1 MB LFAREA that is allocated in megabytes, in decimal, on behalf of pageable 1 MB-page requests.
- The amount of 2 GB LFAREA that is allocated in 2 GB units, in decimal.
- The high water mark for the amount of 2 GB LFAREA that is allocated in 2 GB units, in decimal.

> **Important:** Changes to the IEASYSxx LFAREA parameter require an IPL.

If the LFAREA is defined, DB2 automatically tries to use the 1 MB frames for buffer pools defined with PGFIX(YES) by default. Starting with DB2 11, you can override the default by specifying FRAMESIZE(4 K) or FRAMESIZE(2 GB) for the DB2 buffer pool. You can also explicitly specify FRAMESIZE(1 MB). 2 GB frames exploitation requires a System z zEC12 processor and requires the buffer pool to be non-pageable (PGFIX(YES)).

Lab experiments showed equivalent CPU use for the workload when using 2 GB frame compared to 1 MB frame. So currently, performance advantages of 2 GB frames have not been observed. However, it is expected that as memory sizes become much larger, 2 GB frames may become advantageous. The minimum number of DB2 buffers to use 2 GB frames is 512 buffers. If the buffer pool size is 3 GB, DB2 allocates one 2 GB frame and 1000 1 MB frames. This can be verified by displaying the buffer pool. There is a potential benefit from using larger frame size by frequently used buffer pool.

Example 3-3 shows the results of **`-DISPLAY BUFFERPOOL`** DB2 command.

Example 3-3 Results of DISPLAY BUFFERPOOL command showing 2 GB and 1 MB frame allocation

```
DSNB401I  -DB1D BUFFERPOOL NAME BP4, BUFFERPOOL ID 4, USE COUNT 1
DSNB402I  -DB1D BUFFER POOL SIZE = 600000 BUFFERS  AUTOSIZE = NO
           VPSIZE MINIMUM = 0 VPSIZE MAXIMUM = 0
             ALLOCATED       =   600000   TO BE DELETED   =        0
             IN-USE/UPDATED  =        0
DSNB406I  -DB1D PGFIX ATTRIBUTE -
             CURRENT = YES
             PENDING = YES
           PAGE STEALING METHOD = LRU
DSNB404I  -DB1D THRESHOLDS -
            VP SEQUENTIAL    = 80
            DEFERRED WRITE   = 30   VERTICAL DEFERRED WRT  =  5,  0
            PARALLEL SEQUENTIAL =50   ASSISTING PARALLEL SEQT=  0
DSNB546I  -DB1D PREFERRED FRAME SIZE 2G
        524288 BUFFERS USING 2G FRAME SIZE ALLOCATED
DSNB546I  -DB1D PREFERRED FRAME SIZE 2G
```

```
       19200 BUFFERS USING 1M FRAME SIZE ALLOCATED
DSNB546I  -DB1D PREFERRED FRAME SIZE 2G
       56512 BUFFERS USING 4K FRAME SIZE ALLOCATED
```

3.2.2 Flash Express and pageable large pages

The System z zEC12 server supports an optional hardware feature called *Flash Express memory cards*. These memory cards are supported in an I/O drawer with other I/O cards. The cards come in pairs for improved availability and no hardware configuration definition (HCD) or input/output configuration program (IOCP) definition is required. Each card pair provides 1.4 TB storage with a maximum of four card pairs (5.6 TB). Flash memory is assigned to partitions as storage-class memory (SCM). Like main memory, each partition's flash memory is isolated. You can dynamically increase the maximum amount of flash memory on a partition and you can dynamically configure flash memory into and out of the partition.

With the combination of Flash Express installed on a zEC12 and the pageable 1 MB large page support in z/OS V1R13, DB2 exploits the large page support by allocating *buffer control blocks* (PMBs) using 1 MB pageable storage. PMBs consume only 4 - 5 % of buffer pool allocations but are referenced very heavily. They do not need to be page fixed since they are accessed without I/Os. This makes pageable 1 MB large page a natural fit for PMB. These large pages may be paged to and from Flash Express and performance may be improved due to reduction in TLB misses and increase in the TLB hit rate.

Performance enhancement

The brokerage online transaction processing (OLTP) workload is exercised to demonstrate the performance benefit of having PMBs residing on pageable 1 MB pages.

In the following set of measurements, all buffer pools are on 4 KB frame and PGFIX=YES. When neither SCM or LFAREA is available, PMBs are allocated to 4 KB pageable frame (base measurement). With the availability of SCM, PMBs are allocated to 1 MB pageable frames, taking advantage of reduction in TLB misses and an increase in the TLB hit rate.

Laboratory tests showed 2.1% CPU time reduction in both DB2 class 2 and total DB2 CPU time with PMBs on 1 MB pageable frame. Measurement results are summarized in Table 3-2.

Table 3-2 Measurement results for PMB on 4 KB versus 1 MB pageable frame

CPU time in msec. per COMMIT	PMBs on 4 KB pageable frame	PMBs on 1 MB pageable frame	Improvement in %
Class 2	1.546	1.514	-2.1
MSTR	0.005	0.002	
DBM1	0.064	0.061	
IRLM	0.007	0.011	
Total	1.622	1.588	-2.1

Regardless of whether Flash Express is installed, paging with any of DB2 storage pools is strongly discouraged. Pageable 1 MB pages need to be backed by real storage to achieve performance improvements.

This function is effective in DB2 11 Conversion Mode. No user action is required to enable this function from the DB2 side.

Hardware and software requirements

This feature requires zEC12 (2827) hardware with Flash Express installed, as well as z/OS V1R13 and above with requisite PTFs (FMID JBB778H). APARs PM85944 and PM90486 retrofit this feature to DB2 10 for z/OS.

Useful information for storage-class memory

Use the **DISPLAY ASM** command to identify the page data sets and SCM blocks that the system is currently using. You can request this information for each data set of a given type, or for a specific data set.

If you request information by data set type, information is displayed in message IEE200I for each data set of the specified type that the system is currently using.

Example 3-4 shows the helpful information regarding SCM and auxiliary storage manager (ASM) as displayed by commands.

Example 3-4 Displaying SCM and ASM information

```
      IEASYSxx:  PAGESCM=ALL     (default)

D M=SCM
IEE174I 18.45.20 DISPLAY M 999
STORAGE-CLASS MEMORY STATUS
768G DEFINED
ONLINE
 0G-768G
0G OFFLINE-AVAILABLE
0% IN USE
SCM INCREMENT SIZE IS 16G


D ASM
IEE200I 18.45.21 DISPLAY ASM 005
TYPE     FULL STAT   DEV  DATASET NAME
PLPA      39%   OK  E8C5  SYS1.STLAB7C.PLPA.PAGE
COMMON     0%   OK  E8C5  SYS1.STLAB7C.COMMON.PAGE
LOCAL      0%   OK  E8C5  SYS1.STLAB7C.LOCAL.PAGE
SCM        0%   OK   N/A   N/A
PAGEDEL COMMAND IS NOT ACTIVE

D ASM,SCM
IEF196I IEF237I EB6B ALLOCATED TO SYS00036
IEE207I 11.22.46 DISPLAY ASM 006
STATUS      FULL              SIZE             USED         IN-ERROR
IN-USE        0%       201,326,592           14,268                0
```

3.3 RoCE network adapters

The zEnterprise RoCE-Express adapter provides a Remote Direct Memory Access (RDMA) capable network adapter (RDMA network interface card or RNIC) enabling access to RDMA over Converged Ethernet (RoCE) networks via standard 10 GbE switches. With the zEnterprise RoCE-Express adapter solution, you get improved transaction rates for transactional workloads due to improved network latency and lower CPU cost for workloads with larger payloads while preserving the key qualities of services required by System z clusters in enterprise data center networks. Using *shared memory communications* over RDMA (SMC-R) and a new sockets-based communications protocol, network latency is improved versus 10 Gbit Ethernet.

RoCE uses a reduced networking stack with streamlined, low level, RDMA interfaces, without any involvement by the remote host's CPU when writing to partner memory, combining to provide potentially significant CPU savings versus standard Ethernet networks.

SMC-R is also designed to provide IBM HiperSockets™ like performance across Ethernet (RoCE) for multiple CPCs. SMC-R is supported for any LPAR-to-LPAR configuration (same or different CPCs). Within a single CPC, HiperSockets could still be the best price and performance option. Also note that multiple LPARs cannot share a single 10 GbE RoCE Express feature. If your CPC has Ethernet LAN access to another CPC, you are a candidate to benefit from SMC-R.

For details, see the following website:

http://www-01.ibm.com/support/docview.wss?uid=tss1fq131485&aid=1

3.3.1 Requirements

Each RoCE-Express adapter is dedicated to a single LPAR. The best practice is to use two adapters per LPAR for redundancy. An existing Ethernet infrastructure is required and z/OS Communications Server SMC-R support in z/OS 2.1 is required. zEC12 GA2 upgrade is required. Communication between LPARs uses JCC Type 4 driver.

No application changes are required.

Note: Refer to *IBM System z Connectivity Handbook,* SG24-5444-13 for more information.

3.3.2 Performance measurements

The following performance measurements were conducted to demonstrate the increased transaction rates provided by the reduced network latency of RoCE-Express. The measurements were done using a zEC12 processor with three dedicated processors, two 10 GbE adapters, two RDMA capable network interface cards (RNICs), z/OS V2R1, GA2 level firmware. We measured the following workloads:

- The IRWW distributed workload
- A large fetch workload

The IRWW distributed workload

The distributed version of the IRWW JDBC workload was used as an example of a simple OLTP-like application. The JDBC application runs under z/OS UNIX System Services on one LPAR and communicates via SMC-R to DB2 for z/OS on another LPAR.

Table 3-3 on page 74 summarizes the performance observations.

Reduced network latency improves the throughput of this workload by decreasing the time spent in the network and improving concurrency (less time in the network means less time holding locks).

Comparing RoCE with OSA 4S 10 Gbit, notice the 39% reduction in class 1 elapsed time, which leads to a 66% ETR improvement at no CPU increase.

RoCE-Express uses less CPU than HiperSockets in this configuration and can also support z/OS LPARs on different machines.

Table 3-3 Reduced network latency with IRWW distributed workload

Adapter type	ETR	ITR	CL1 ET	CL 2 ET	CL1 CPU	CL2 CPU	z/OS CPU%
OSA 1 Gbit TCP/IP	1787.06	3827.49	0.008129	0.002734	0.000571	0.000329	46.69
OSA 4S 10 Gbit TCP/IP	1869.92	3847.57	0.007779	0.002659	0.000570	0.000326	48.60
RoCE 10 Gbit SMC-R	3108.94	3845.32	0.004749	0.002763	0.000572	0.000333	80.85
HiperSocket	2931.44	3355.59	0.005049	0.002758	0.000680	0.000342	87.36

A large fetch workload

This other performance measurement was conducted to investigate the CPU reduction provided by using RoCE-Express. The “Large Fetch Workload” was used for measuring the CPU improvement seen by using the SMC-R over RoCE solution. The Large Fetch Workload executes a single query that returns 500,000 rows of 32 KB each. The query executes three times in a loop. The QRYBLKSZ was set to 4 MB. Hipersockets is not reported because it did not work at the time for that blocksize.

The bandwidth utilization is 54 MB/sec with the 10 Gbit OSA and 85 MB/sec with RoCE.

The results are shown in Table 3-4.

RoCE shows a 37% response time (CL1 ET) improvement on the server and requester side versus 10 Gbit OSA.

The DB2 CPU time reduction is 26% on the server side with a 57% decrease in z/OS CPU. A 33% decrease in z/OS CPU is seen on the requester side.

The CPU savings come from a streamlined network stack as well as the lack of host CPU intervention when writing to a partner's memory.

Table 3-4 CPU reduction provided by using RoCE-Express

Adapter type	Requester			Server		
	CL1 ET	CL1 CPU	Total CPU	CL1 ET	CL1 CPU	Total CPU
OSA 1 Gbit TCP/IP	1365.71	45.91	217.44	341.42	9.39	29.29
OSA 4S 10 Gbit TCP/IP	291.86	45.88	70.63	72.95	9.87	43.31
RoCE 10 Gbit SMC-R	183.37	45.57	46.83	45.96	7.64	19.03

3.4 1 MB code pages

The ability to execute code loaded into 1 MB pageable frames is automatically set by DB2 11 with z/OS 2.1 when Flash Express is configured. Since other enhancements are enabled with z/OS 2.1, it is not possible to separate just the 1 MB code pages improvement. See 11.1.2, “DB2 related enhancements in z/OS 2.1” on page 249.

The estimate is about 2% CPU reduction.

3.5 200,000 open data sets

The DSMAX subsystem parameter determines the maximum number of data sets that DB2 is allowed to keep open at one time in the DBM1 address space. In DB2 10, the maximum value for DSMAX was 100,000. DB2 11 increases this maximum number to 200,000. This support is available in DB2 11 CM mode for z/OS 1.13 and later versions and it was retrofitted to DB2 10 via APAR PM88166. The APAR is also available in DB2 10 CM8 or CM9 mode. Use z/OS Version 1 Release 13 or higher for any DB2 subsystem that will have a DSMAX setting greater than 100,000.

Important: The practical limit of open data sets can be less depending on available storage below the 2 GB bar. You should allow for 1 GB of virtual storage below the 2 GB bar for 200,000 open data sets.

The measurements in this section show how the performance is affected as the number of open data sets increases to 200,000 with z/OS 1.13.

The measurements were conducted to observe the storage, CPU, and open data set performance impact opening 200,000 data sets with DB2 11 using z/OS 1.13 on a zEC12 processor with 3 CPs. Measurements are done with MEMDSENQMGT enabled, GRS STAR and enhanced catalog sharing turned on.

Figure 3-1 on page 76 shows the *total real storage* usage on 31 bit and 64 bit. The measurements show a steady increase in storage usage as DB2 approaches DSMAX 200,000 topping out at 991.46 MB of 31-bit storage, 614.53 MB of 64-bit storage for a total of 1,606 MB of total real storage used.

With 20 concurrent SQL jobs executed to open 200,000 data sets, it took 16 - 18 minutes to open all of the data sets. The measured total CPU time to open 200,000 data sets with z/OS 1.13 on zEC12 processor was 540 seconds.

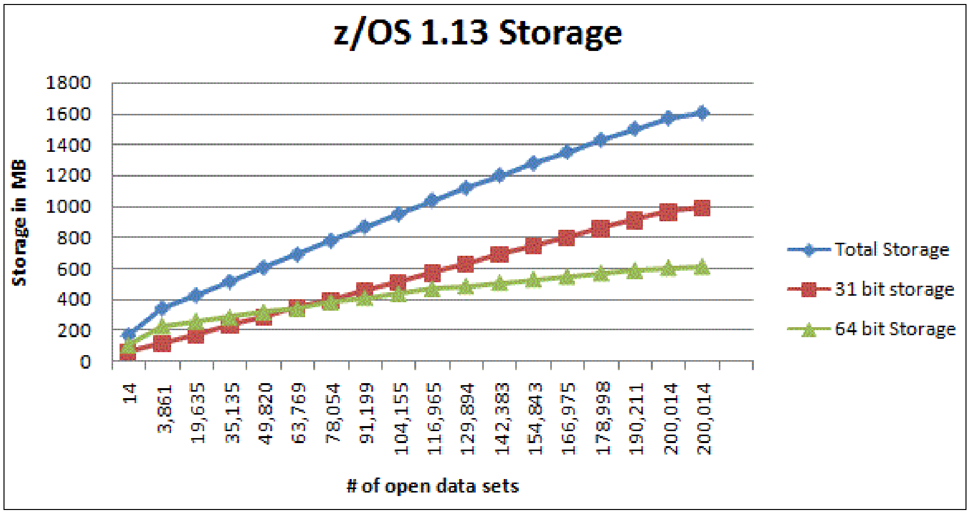

Figure 3-1 Real storage use with z/OS 1.13

3.6 Query parallelism storage-related reduction

The DB2 subsystem parameters CTHREAD and MAXDBAT limit the number of threads and distributed SRBs, but there is no cap for parallel child tasks SRBs. When there are too many threads or SRBs running on the system, below the bar storage or ECSA storage may reach to the limit and cause the z/OS system to abend. Although parallelism-related storage moved above the bar since DB2 10, the storage below the bar can still be large if the query is complex and parallel degree is high.

DB2 11 provides an infrastructure with no user external parameters to automatically reduce the number of parallel threads at execution time if there is shortage of storage below the bar or ECSA storage.

The infrastructure that is provided in DB2 11 (also in DB2 10 with APAR PM24723) provides the following levels:

Level 1 OK: Query runs with planned parallel degree

Level 2 Mild warning: Reduce parallel degree by ¼

Level 3 Moderate warning: Reduce parallel degree by ½ or to degree 2

Level 4 Severe warning: Reduce to sequential run

Level 5 Melt down: Reduce to sequential run

A new value QW0221C8 is added for IFCID 221 QW0221RN field to indicate that the parallel group ran on reduced degree or degenerated because of system storage shortage. The description of PM24723 states IFCID 225 real storage statistics enhancements.

3.7 Exploitation of large memory for buffer pools

Using the large memory can potentially improve both elapsed time and CPU time. In this section, we describe the benefit of using large buffer pools in DB2. The brokerage online transaction workload is used for the evaluation.

3.7.1 Measurements description

We varied the size of the buffer pool to observe the effect on performance of OLTP workload. Buffer pools were always backed up by 1 MB frames and there was enough real storage defined in the LPAR to avoid paging operation. The workload used the following configuration:

- Hardware
  - EC12 system with 32 CPs
  - 240 GB real memory with 180 GB LFAREA for 1 MB frames
  - DS8870
- z/OS 1.13
- DB2 11
  - Total buffer pool sizes 10 GB, 24 GB, 70 GB, 160 GB with 1 MB frame size and PGFIX(YES)
- Workload (Brokerage workload)
  - 1.3 TB of data
  - 13 buffer pools
  - 33 table spaces
  - 68 indexes

All measurements were taken after a steady state of throughput was achieved.

Figure 3-2 shows the effect of larger buffer pools (10 - 70 GB) for this workload. The benchmark workload is built in such a way that the faster the transactions are processed, the more transactions are triggered. As we increased the buffer pool size, we observed response time reduction and higher throughput.

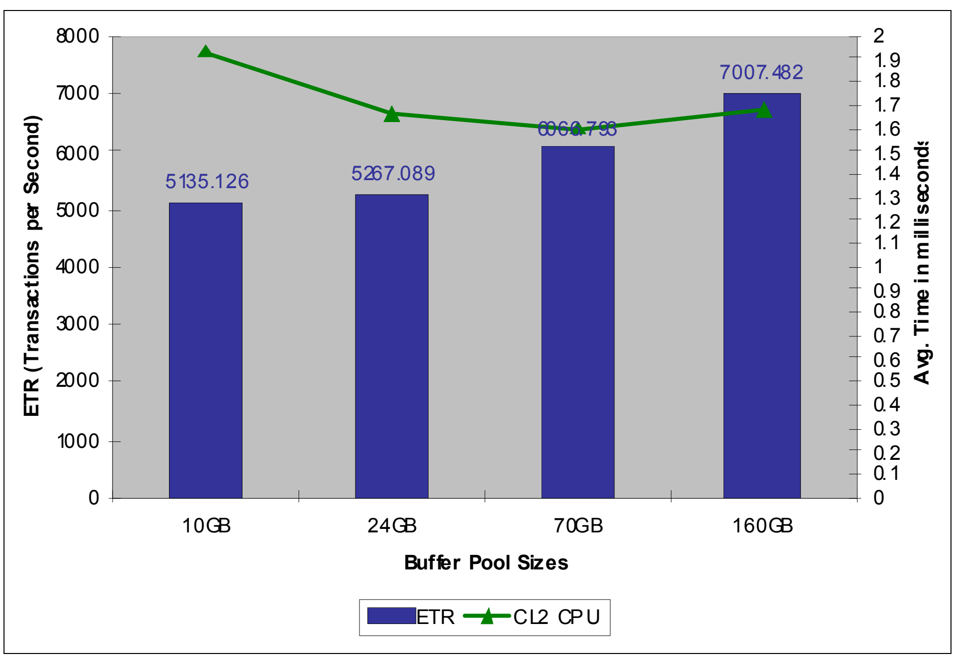

Figure 3-2 Average CL2 CPU versus throughput with increased storage

Increasing the total buffer pool size from 10 to 24 GB showed 3% response time reduction by reducing database I/O requests from 26 to 13 synchronous I/Os per transaction. The transaction response time did not reduce as much as reduction of number of I/O waits. This is because most of the database I/O that we eliminated had been previously satisfied from storage subsystem cache. On the contrary, DB2 CPU time shows 14% reduction. This 14% CPU reduction comes from avoiding CPU cycles on I/O processing.

Increasing the buffer pool size from 24 to 70 GB further reduced the synchronous I/O requests. This results in further response time reduction as well as CPU reduction. However, when additional buffer pools are added to 160 GB, it eliminated only one sync I/O per transaction. This sync I/O request was not satisfied from storage subsystem cache, thus it still helped to reduce the transaction response time further. On the CPU side, reducing one I/O did not show any visible reduction and slight CPU increase was observed with the case of 160 GB compared to 70 GB case. The CPU increase can be attributed to throughput increase.

As the number of transaction increases, we observed more getpages per transaction as the database continues to age quicker while there is minimum CPU reduction from additional buffer pools. This behavior is specific to the benchmark. Larger buffer pools continue to help response time reduction, but we reached the point of diminishing return at 70 GB buffer pool size in term of CPU reduction.

Although DB2 I/O dropped with buffer pool size increase, the average response time per I/O increased. This is because as the buffer pool size increases, I/Os are less likely to be satisfied from storage subsystem cache. Although the log write I/O remained the same, the average response time per log write I/O increased with buffer pool size for this measurement. We believe this is because the logs and data share physical storage devices and therefore there is more contention as throughput increases.

Figure 3-3 shows that as the buffer pool size increases, throughput increases, and database I/O decreases while log write I/O stays the same.

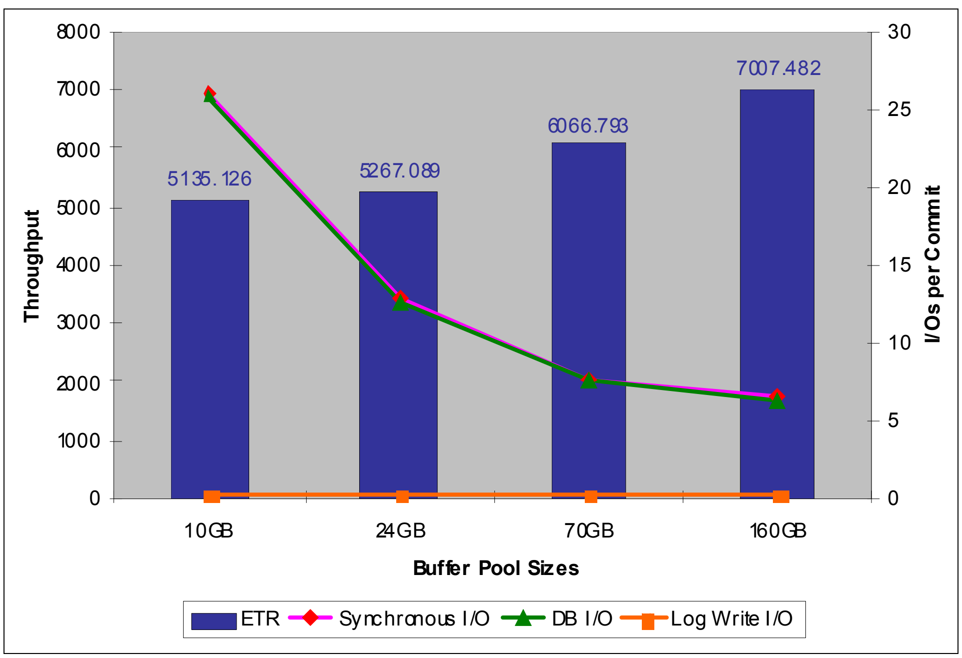

Figure 3-3 Throughput versus synchronous I/O

3.7.2 Considerations or usage guide

The buffer pools should be fully backed up by real storage to avoid any negative impact from paging. Before altering the size of a buffer pool, monitor the buffer pool statistics. If there is little I/O activity for a buffer pool, there is not much advantage to increasing the buffer pool size. If there is I/O activity to the buffer pool with very few getpages, increasing the size of the buffer pool might not buy you much CPU reduction since its contribution to overall CPU usage is already minimal. But, it might still increase the throughput. DB2 statistics class 8 or IFCID 199 can help to identify the object with many synchronous I/O requests.

The benefit of using large buffer pool highly depends on the workload. One of the factors is the data access pattern. If the access pattern is random, a larger buffer pool should continue to help to reduce sync I/Os, response time, and CPU until entire objects fit in the buffer pools. If objects are sequentially accessed, there is still response and CPU time reduction but the time spent in sequential I/Os is zIIP eligible and CPU savings might not translate directly to cost reduction.

The size of the working set is another factor. Regardless of the size of the entire object, if the working set is relatively small, a small buffer pool can satisfy most requests. In this example, the workload had 1.3 TB of data and indexes, but at 70 GB we reached the optimal buffer pool size in term of CPU savings.

For more information about buffer pool tuning, see *DB2 11 for z/OS Buffer Pool Monitoring and Tuning*, REDP-5092.

4

Availability

DB2 11 offers the following set of functions, which help the availability of DB2 systems and applications:

- Online schema changes and enhanced recovery options
- Automatic recovery of indexes from GRECP or LPL status
- Improved availability when altering limit keys
- Work file database enhancements
- Governing of parallel processing of utilities
- Compression dictionary availability for CDC tables
- DROP column support
- Defer define object enhancements
- Allow BIND, REBIND, and DDL to break-in persistent threads
- Idle thread break-in

See the chapter “Availability” of *DB2 11 for z/OS Technical Overview*, SG24-8180.

In this chapter, we focus on how, under some conditions, DB2 11 can autonomically minimize the impact on performance of disorganized data and therefore reduce the need for running the REORG utility. These improvements come in the following ways:

- By automatically cleaning up pseudo-deleted index entries
- By managing free space to reduce the number of indirect references when the row size varies

This chapter contains the following sections:

- Introduction
- Pseudo-deleted index key cleanup
- Indirect references reduction

4.1 Introduction

The online REORG utility reorganizes a table space and index space to improve access performance, reclaim fragmented space, and materialize changes. DB2 11 introduces several improvements to online REORG:

- Improve performance of partition-level REORG with nonpartitioned secondary indexes
- SWITCH phase impact reduction
- Physically delete empty partition-by-growth partitions
- Automated REORG mapping table management
- REORG without SORTing data
- Partition-level inline image copy
- Improved REORG LISTDEF processing
- REBALANCE enhancements
- REORG of LOB enhancements
- Improved REORG serviceability
- REORG change of defaults to match best practices

For a description of these changes, refer to *DB2 11 for z/OS Technical Overview*, SG24-8180. For performance data, see Chapter 10, “Utilities” on page 229.

Despite these improvements, REORG remains a potentially costly operation that often involves some degree of unavailability. Today’s environments require a high level of data availability and application offline windows tend to disappear. So there is a general interest on reducing the need for REORG.

Hardware innovations introduced in enterprise storage systems during the last years contribute to reduce the need for REORG. Solid-state disks (SSDs), for example, have no mechanical seeks or rotational delays that are associated with disorganized data, enabling these devices to stream the data efficiently no matter how the data is organized. The High Performance IBM FICON® (zHPF) and DS8000 6.2 have improved support for DB2 prefetch and list prefetch. For details, see *DB2 for z/OS and List Prefetch Optimizer,* REDP-4862.

DB2 10 for z/OS also decreased the need for REORG with two enhancements:

- List prefetch to perform disorganized index scan
- Row level sequential detection (RLSD)

DB2 11 is the next step in the evolution towards meeting the goal of reducing the need for REORG. Two features of DB2 11 move in this direction and provide you with more consistent performance. They are described in this section:

- Pseudo-deleted index key cleanup
- Indirect references reduction

While reading this information, keep in mind that some changes require REORG for materialization, such as converting from basic to the extended RBA/LRSN format of an existing page set or changing some of the table space or index attributes.

4.2 Pseudo-deleted index key cleanup

When a row is deleted, the *index entry*[1] for the key to the row must be removed in each index. However, DB2 cannot immediately delete the index entry, unless it has exclusive access control because of the possibility of a rollback. Instead, DB2 sets a bit in the index to mark the index entry as being *pseudo-deleted*.

These pseudo-deleted index entries occupy space in the index. Pseudo-empty index pages are pages that contain only pseudo-deleted index entries. With an accumulation of pseudo-deleted index entries, SQL index access performance may degrade over time.

There is a performance impact for maintaining index pseudo delete entries. SQL operations such as SELECT, FETCH, UPDATE, or DELETE that require an index search could result in more getpages and more lock requests to access the required data. INSERT, UPDATE, and DELETE operations may see concurrency issues. There could be collisions with committed pseudo-deleted index entries. Also, record identifier (RID) reuse by an INSERT statement following a DELETE statement could cause a deadlock.

Before DB2 11 for z/OS, pseudo-deleted index entries are not effectively cleaned up and tend to remain until a REORG of the index cleans them up. A REORG of the table space or partition also cleans up the pseudo-deleted index entries. REORG may be necessary to improve SQL index access performance back to an acceptable level.

In DB2 11, DB2 autonomically deletes pseudo-empty index pages and pseudo-deleted index entries by scheduling asynchronous service tasks to clean up committed pseudo-deleted index entries. In the case of a pseudo-empty index page, the whole page can be removed from the index tree. By limiting the accumulation of the pseudo-deleted index entries, SQL index access performance can remain at an acceptable level. This function is designed to remove committed pseudo-deleted entries from the indexes with minimal or no disruption to other concurrent DB2 work in the system.

The service task overhead is not associated with any DELETE or UPDATE activity. The cleanup service tasks (daemons) have low CPU overhead. The completed activity for each index page that has pseudo-deleted index entries removed can be tracked by IFCID 377. When the system has been configured with one or multiple System z Integrated Information Processor (zIIP) processors, this automated cleanup function runs under enclave service request blocks (SRBs) that are zIIP-eligible.

This cleanup process occurs only when the index is being updated by DB2 processes such as INSERT, DELETE, or UPDATE. DB2 algorithms decide when and what index pages to clean up based on timer intervals and index statistics collected by Real Time Statistics. Cleanup of pseudo-deleted entries does not occur immediately after COMMIT but only after a period of time the page sets are open for update for other DB2 activity, and the asynchronous service tasks are scheduled to process the cleanup. A threshold of the number of pseudo-deleted index entries per page must also be met to engage the cleanup process. Some pages with few entries may not be processed for cleanup.

The cleanup occurs by default in DB2 11 conversion mode (CM), but you can use these options to limit the amount of cleanup activity, when specific index page sets may be processed, or by disabling it selectively altogether.

[1] A unique index entry is made up by the key value, a flag, and a record ID (RID), which points to a DB2 record (combination of the record prefix and the row). The flag indicates if the RID is marked for deletion, uncommitted, or a gap. For non-unique indexes, each index entry is preceded by a header with the count of the number of RIDs. The RID includes page number, offset within the page, and partition number.

You control the cleanup activity by using:

- INDEX_CLEANUP_THREADS subsystem parameter
- SYSIBM.SYSINDEXCLEANUP catalog table

4.2.1 INDEX_CLEANUP_THREADS subsystem parameter

The new DB2 system parameter INDEX_CLEANUP_THREADS determines the number of threads that are allowed to work on the cleanup of pseudo deleted index entries. This parameter works with the SYSIBM.SYSINDEXCLEANUP catalog table, which controls cleanup processing of pseudo-deleted index entries.

You can specify any value 0 - 128. The default value is 10. Setting this subsystem parameter to 0 disables index cleanup. This parameter is online changeable. The recommendation is to use 10. If the pseudo-deleted entries in RTS are zero, the cleanup is not activated.

If INDEX_CLEANUP_THREADS has a value greater than zero, DB2 checks the real-time statistics to identify the indexes with many pseudo-deleted entries or pseudo empty pages. The columnames in SYSINDEXSPACESTATS are:

- REORGPSEUDODELETES (pseudodelete entries)
- NPAGES (pseudodelete pages)

The real-time statistics are checked periodically to identify the indexes with the most pseudo-deletes entries. An index can only be cleaned up when a thread is freed up, and the index candidates are sorted based on the number of pseudo-deletes, so the ones with the most pseudo-deletes get cleaned up first.

The automated cleanup of pseudo-deleted entries in DB2 11 cleans up both pseudo-empty index pages and pseudo-deleted index entries. The benefits of this process are that it reduces the impact of pseudo delete entries, and it reduces the need to run the **REORG INDEX** utility.

The following potential concerns exist about automated clean up:

- Possible CPU overhead
- Disruption to other concurrent threads
- Increase in log volume introduced by the cleanup process

These concerns can be managed by tuning the maximum number of concurrent threads by setting the system parameter INDEX_CLEANUP_THREADS. In data sharing, each member of the group can use a different setting for INDEX_CLEANUP_THREADS.

Generally the impact is minimal and the default values are a good starting point.

You can use the DB2 11 new IFCID 377 to monitor cleanup activity. Its records include the DBID, PSID of the index, the partition number of the index, and the page number being cleaned up. It has an indicator to show if the cleanup is a pseudo-empty page cleanup, in which case, the pseudo-empty index page is deleted from the index tree, or if the cleanup is pseudo-deleted entry cleanup, in which case, the index page remains in the index tree. It also has a field to show the number of pseudo-deleted entries removed from each index page. IFCID 377 is not included in any trace class because its volume can be large.

Example 4-1 shows the fields in IFCID 377.

Example 4-1 IFCID 377 fields

```
*********************************************************************
*   IFCID 0377 to record index pseudo delete daemon cleanup         *
*********************************************************************
*
QW0377   DSECT                IFCID(QWHS0377)
QW0377DB DS    CL2            DATA BASE ID
QW0377OB DS    CL2            INDEX PAGE SET ID
QW0377PT DS    CL2            PARTITION NUMBER
QW0377FL DS    CL1            FLAGS
         DS    CL1            RESERVED
QW0377PG DS    CL4            PAGE NUMBER of the index page cleaned up
QW0377NU DS    CL4            NUMBER OF PSEUDO DELETED ENTRIES REMOVED
QW0377DL EQU   X'80'          PSEUDO EMPTY PAGE IS DELETED
QW0377CL EQU   X'40'          PSEUDO DELETED ENTRIES CLEANED UP
*
```

Example 4-2 shows a portion of an OMEGAMON XE for DB2 Performance Monitor on z/OS Record Trace Report with IFCID 377 details.

Example 4-2 OMPE Report IFCID 377

```
PRIMAUTH CONNECT   INSTANCE      END_USER      WS_NAME                         TRANSACT
ORIGAUTH CORRNAME  CONNTYPE      RECORD TIME   DESTNO ACE IFC  DESCRIPTION    DATA
PLANNAME CORRNMBR                TCB CPU TIME             ID
-------- -------- ----------- ---------------- ------ --- --- -------------- -------------------------------------------------------
SYSOPR   DB2B     CB6EA03AAC12 N/P             N/P                            N/P
SYSOPR   014.IDAE 'BLANK'     11:47:21.66633960    454   1 377 PSEUDO DELETE  NETWORKID:  DKBDON01  LUNAME:  BDPODTST  LUWSEQ:   144
'BLANK'  MK01                 N/P                              DAEMON CLEANUP
         |------------------------------------------------------------------------------------------------------------------------
         |DATABASE ID       :  403                        INDEX PAGE NUMBER  :          11       PARTITION NUMBER   :          1
         |INDEX PAGE SET ID  :  16                        PD ENTRIES REMOVED :         594
         |FLAG:             :  PAGE IS DELETED FROM INDEX
         |------------------------------------------------------------------------------------------------------------------------
                              11:47:21.67700906    455   1 377 PSEUDO DELETE  N/P
                              N/P                              DAEMON CLEANUP NETWORKID:  DKBDON01  LUNAME:  BDPODTST  LUWSEQ:   144
         |------------------------------------------------------------------------------------------------------------------------
         |DATABASE ID       :  403                        INDEX PAGE NUMBER  :          34       PARTITION NUMBER   :          1
         |INDEX PAGE SET ID  :  16                        PD ENTRIES REMOVED :         594
         |FLAG:             :  PAGE IS DELETED FROM INDEX
```

4.2.2 SYSIBM.SYSINDEXCLEANUP catalog table

Automate cleanup of pseudo-empty index pages and pseudo-deleted index entries is enabled for all indexes by default when the value of the INDEX_CLEANUP_THREADS subsystem parameter is set to a nonzero value. However, if you decide it is needed, you can specify time windows to enable or disable the index cleanup for the entire subsystem, for indexes in specific databases, or for specific indexes. This is done by inserting rows in the SYSIBM.SYSINDEXCLEANUP catalog table. You can use this table to specify the time when indexes are subject to cleanup. It indicates when and which indexes are enabled or disabled for cleanup.

The catalog table includes the following information for use in the cleanup process:

- Name of databases and indexes
- Cleanup enabled or disabled
- Day of week or day of month
- Start time and end time

The layout of SYSIBM.SYSINDEXCLEANUP, with a short description of its columns, is shown in Table 4-1.

Table 4-1 SYSIBM.SYSINDEXCLEANUP layout

Column name	Description
DBNAME	The name of the database that contains the index space.
INDEXSPACE	The name of the index space.
ENABLE_DISABLE	Specifies whether the row enables or disables cleanup for the specified index space. 'E' Enabled 'D' Disabled
MONTH_WEEK	Indicates the meaning of the value of the DAY column: 'M' The value indicates the day of the month. 'W' The value indicates a day of the week.
MONTH	The months in which the time window applies. For example, a 1 value indicates January and a 12 value indicates December. If this column contains NULL, the time window applies to all months. If the value of the MONTH_WEEK column is 'W', this value must be NULL.
DAY	The day of the month or the day of the week for which the time window applies, as specified by the value of the MONTH_WEEK column. For example, if MONTH_WEEK='W', a 1 value indicates Monday and 7 indicates Sunday. If the value of this column is NULL, the time window applies to every day of the month or every day of the week.
START_TIME	The local time at the beginning of the time window specified by the row. When this column contains a null value, the row applies at all times on the specified days. This column must contain NULL if the END_TIME column contains NULL.
END_TIME	The local time at the end of the time window specified by the row. When this column contains a null value, the row applies at all times on the specified days. This column must contain NULL if the START_TIME column contains NULL.

When there is an index that needs to be cleaned up, DB2 checks the SYSIBM.SYSINDEXCLEANUP catalog table to see if entries in this table allow this index to be cleaned up at the current time. If the SYSIBM.SYSINDEXCLEANUP catalog table is not accessible, index cleanup is disabled; no index can be cleaned up in the system.

DB2 checks the data in the SYSIBM.SYSINDEXCLEANUP table at 10-minute intervals. In consequence, the enforcement of any new row that you insert might be delayed for as long as 10 minutes. To stop index cleanup processing immediately, set the value of the INDEX_CLEANUP_THREADS subsystem parameter to 0.

Tip: Use rows in the SYSIBM.SYSINDEXCLEANUP only to define exceptions when the default index cleanup behavior is unacceptable.

The performance of index cleanup processing might be reduced if the number of rows in this catalog table becomes too large. To minimize the performance impact:

- Whenever possible, define time windows at system or database levels, rather than for specific indexes
- Remove unneeded or conflicting rows from the catalog table

Refer to *DB2 11 for z/OS Technical Overview*, SG24-8180 for examples of using SYSIBM.SYSINDEXCLEANUP.

4.2.3 Performance measurements

With DELETE, indexes show an increased number of pseudo-deleted entries between REORG boundaries if no pseudo-deleted key cleanup is ongoing. The WebSphere Application Server Portal workload is used to illustrate this process, and the performance impacts that this can produce.

IBM WebSphere Portal products provide enterprise web portals that help companies deliver a highly-personalized, social experience for their customers. WebSphere Portal products give users a single point of access to the applications, services, information, and social connections they need. These products help increase visitor response and reduce web operations cost while offering a range of capabilities to meet your business needs. In the measurements, Portal 6.1.5 was used.

The default values of INDEX_CLEANUP_THREADS and SYSIBM.SYSINDEXCLEANUP were used.

The changes in pseudo-deleted entries were monitored using the information collected in the Real Time Statistics table SYSIBM.SYSINDEXSPACESTATS. Table 4-2 lists the columns that provide information about pseudo-delete index entries.

Table 4-2 SYSINDEXSPACESTATS columns with pseudo-delete related information

Column name	Description
NPAGES	The number of pages in the index tree that contain only pseudo-deleted index entries.
REORGPSEUDO-DELETES	The number of pseudo-deleted index entries stored in the index space or partition. A pseudo-delete is a RID entry that has been marked as deleted. A null value indicates that the number of pseudo-deleted index entries is unknown.

The workload was executed to simulate the activity of five business days. Measurements are done under one IBM z10™ LPAR with 4 CPs using z/OS 2.1. DB2 is configured as non-data sharing. We started as DB2 10 NFM, took the base measurements, then migrated the workload to DB2 11 CM. We used 50 concurrent clients to drive the requests to access the website.

Figure 4-1 on page 88 illustrates the measurements of the WebSphere Application Server Portal workload over a period of five days to illustrate the effects of pseudo-deleted index entries over time.

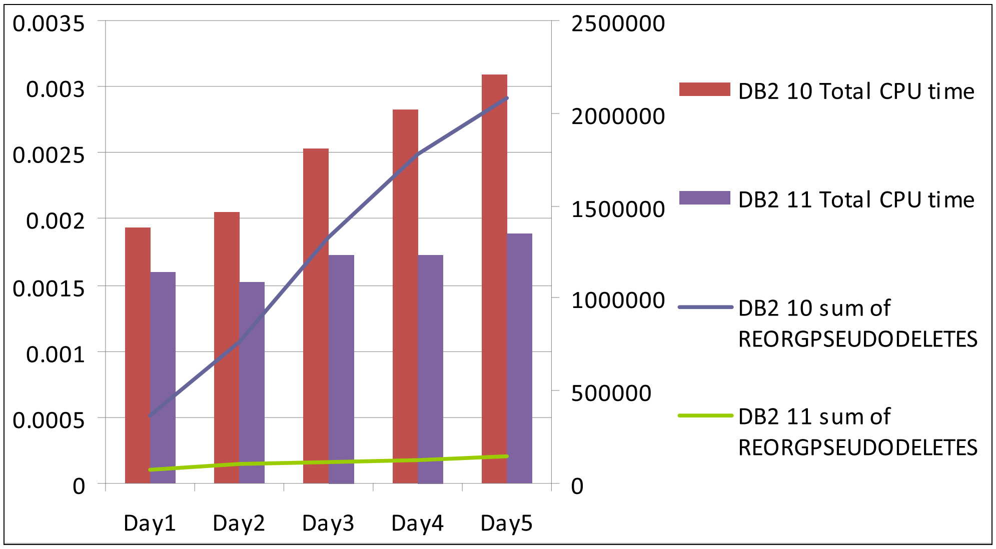

Figure 4-1 WebSphere Application Server Portal workload over a period of five days

The CPU time per transaction grew from slightly under 2 milliseconds to more than 3 milliseconds.

With DB2 10, the number of pseudo-deleted entries increases by about 50,000 every day during the complete test.

With DB2 11, this number remains almost constant during the same period. This chart also shows the evolution of the CPU cost per transaction. With DB2 10, there is a regression of 50%, while with DB2 11, the regression is about 20%.

For this same workload, by the fifth day DB2 11 had reduced the number of pseudo-deleted index entries by 93% compared to DB2 10. While DB2 11 had achieved 20% less CPU time than DB2 10 on day 1, the CPU reduction grew to 39% on day 5.

4.2.4 Conclusions

Indexes subject to DELETE operations show an increase in the number of pseudo-deleted entries and pseudo-deleted pages over time. A high number of pseudo-deleted entries may affect SQL performance in terms of both CPU and elapsed time and can lead to a significant increase in the number of IRLM requests in some inserts and updates.

Before DB2 11, or with DB2 11 with pseudo-deleted index key cleanup disabled, performance degradation is present between REORG boundaries. Pseudo-deleted index key cleanup in DB2 11 allows to extend the period between REORGs by cleaning pseudo-deleted entries and then avoiding the performance degradation previously observed on applications sensible to them.

4.3 Indirect references reduction

When using variable length rows, or when using data compression, if a row is updated and the row size increases but can no longer fit on the original page, DB2 for z/OS finds another page to store the row. It then modifies the original RID to point at the overflow RID, thus creating an indirect reference since every access to the row requires an extra level of indirection to find the row. An indirect reference requires an extra DB2 getpage, which often requires extra synchronous I/O.

Figure 4-2 shows an example of how an update can result in a larger row and cause the row to no longer fit on the same page. These rows need to be relocated to a new page, and a pointer to the new page is placed on the original page.

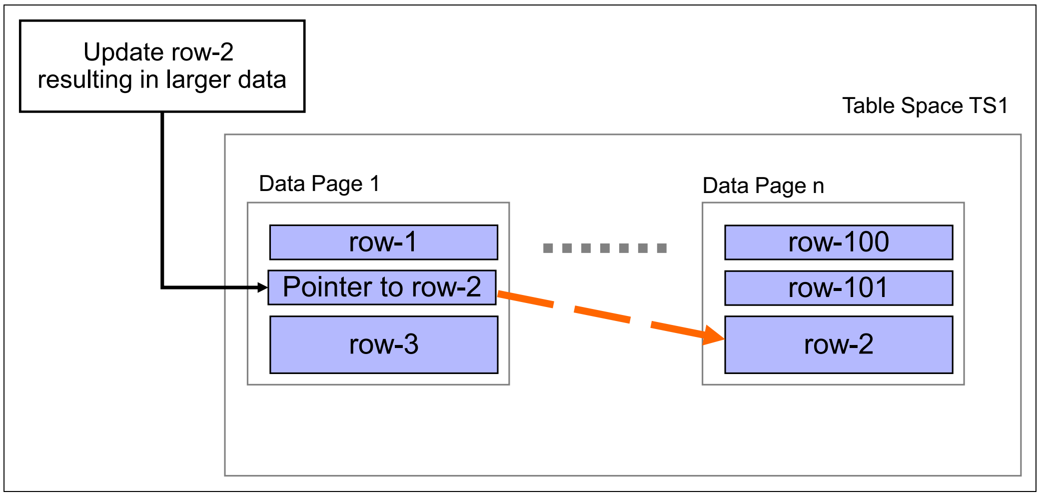

Figure 4-2 Indirect reference: Overflow records

The cost of these overflow records is especially noticeable when DB2 builds a RID list for use with list prefetch because DB2 never overtly prefetches an overflow record. Thus, when DB2 uses list prefetch to some rows that satisfy an SQL predicate, the only synchronous data I/Os are I/Os to read the overflow records.

You can tell if your database contains indirect references by monitoring NEARINDREF and FARINDREF in SYSIBM.SYSTABLEPART:

- NEARINDREF: Number of rows that have been relocated near their original page
- FARINDREF: Number of rows that have been relocated far from their original page

A "near" overflow is one that is likely to be prefetched by dynamic prefetch. However, random row access is more or less equally affected by both "near" and "far" indirect reference.

You can also check REORGNEARINDREF and REORGFARINDREF in the real-time statistics table SYSIBM.SYSTABLESPACESTATS.

The effects on data sharing performance can be as follows:

- With row locksize, a pointer record needs to be locked, and later unlocked because there is no lock avoidance here.
- With page locksize, both a pointer page and an overflow page must be locked, and later unlocked because there is no lock avoidance here.
- Update of GBP-dependent data resulting in an overflow causes synchronous log write I/O wait as a result of DB2 10 PM82279 to fix an incorrect output problem.

Let us consider now the type of applications that are most likely to suffer many indirect references. Without question, nullable VARCHAR columns are indicative of the worst case because some applications insert null values and later update the null values. The greater the update size is as a percentage of the original row size, the more likely it is that indirect references will occur.

Indirect references are also possible when compression is used, simply because a row might not compress as well after an update. Or, a non-null VARCHAR column might grow. But, these situations do not necessarily cause indirect references in a systematic fashion the way nullable VARCHAR columns do.

REORG not only cleans up existing indirect references. REORG can also re-establish more free space via PCTFREE. PCTFREE is the percentage of space on each page that REORG I reserves. LOAD REPLACE also establishes such free space. The reserved space is used by both inserts and updates. The inserts use it to maintain clustering. There is nothing in DB2 10 to prevent the inserts from consuming all of the free space, leaving no reserved space for the updates to increase the row size. Consequently, cluster ratios and indirect references tend to be correlated. If the cluster ratio is high, there will not be many indirect references. When the reserved space becomes exhausted, the cluster ratio starts to degrade and updates that increase the row size start to cause indirect references.

It is also true that when the reserved space is used up, the inserts start to become sequential. Thus, new rows are appended to the end of the table. If those newly inserted rows are also updated in the same order that the rows were inserted, the overflows are sequential too, although the new rows and the overflows could be interspersed among each other. This becomes important when we start talking about dynamic prefetch and sequential prefetch, but it is not important when we talk about random fetch or list prefetch.

MAXROWS is the only tuning option in DB2 10 to avoid indirect references. If MAXROWS is based on the maximum row size, there will never be any indirect references. Alternatively, if MAXROWS is based on the average row size, indirect references will usually be avoided. The success of MAXROWS depends on the row size distribution being somewhat static. If the new rows that are created on Tuesday are of a different size than the rows that were created on Monday, it is hard to choose an optimal MAXROWS value that can apply to both days. Such dynamically changing distributions are unlikely, nevertheless, using MAXROWS requires some performance monitoring and most customers choose not to do it. A more autonomic solution is desirable. That is what DB2 11 provides.

4.3.1 PCTFREE FOR UPDATE and PCTFREE_UPD in DB2 11

DB2 11 provides a new subsystem parameter called PCTFREE_UPD (PERCENT FREE FOR UPDATE field in install panel DSNTIPB), which can be set to AUTO if you would like the inserts to automatically determine how much space to reserve on each page for updates. The default in DB2 11 does not reserve any space for updates.

You can also modify your DDL to specify PCTFREE_FOR_UPDATE. A value of -1 does the same thing as PCTFREE_UPD=AUTO. Or, you can specify a specific percentage as shown in Example 4-3.

Example 4-3 Create/Alter table space FOR UPDATE

```
CREATE/ALTER  TABLESPACE   PCTFREE x  FOR UPDATE y
```

In this example:

- x is % of free space to leave in each data page by LOAD or REORG
- y is % of free space to leave in each data page by INSERT, LOAD, or REORG

An INSERT statement preserves the value provided by y% while REORG preserves (x+y) %. The sum of the values for PCTFREE smallint and FOR UPDATE smallint must be less than or equal to 99.

Furthermore, for a table that was migrated from DB2 10 to DB2 11, or for a table that was created in CM10 mode, a REORG is necessary to reinitialize some of the RTS statistics, namely REORGINSERTS, REORGUPDATES, and REORGDELETES. DB2 uses those statistics to determine how much space to reserve for updates when you plan to use the AUTO setting.

DB2 11 NFM adds a new RTS statistic called UPDATESIZE. UPDATESIZE is the total number of bytes that updates added to existing rows since the last REORG or CREATE. The autonomic algorithm used with PCTFREE_UPD=AUTO uses UPDATESIZE. UPDATESIZE is zero when DB2 is first migrated to NFM. Consequently, it will not be consistent with the older RTS statistics until you run REORG. REORG initializes all of these statistics to zero. If you do not run REORG, the inserts eventually begin to reserve some space, but REORG enables this to happen more quickly.

You should also consider that using PCTFREE_UPD=AUTO or using PCTFREE_FOR_UPDATE -1 on your DDL could cause DB2 to use more DASD space. You can monitor NPAGES to determine the number of pages used for the table space. In typical situations where updates are increasing the row size, allow for the possibility that the autonomics will cause the table space to grow by 10% to 20%. In some cases, you might be able to reduce the amount of DASD space used without having indirect references if you were willing to tune MAXROWS to an optimal value.

4.3.2 IRWW measurements

The IBM Relational Warehouse Workload (IRWW) workload was used to evaluate indirect reference avoidance. This workload is described in Appendix B.1, "IBM Relational Warehouse Workload" on page 298 except that here a "longevity" version was run for many hours. Before beginning the measurements, all of the tables were reorganized to eliminate all indirect references, establish 20% free space (using the old PCTFREE parameter), and rebuild the compression dictionaries. Then, the workload was allowed to run for many hours.

Since the tables were compressed, the row lengths varied. One of the tables called ORDERLINE contained a nullable TIMESTAMP column. New-Order transaction of the workload inserted new rows with null time stamps and the Delivery transaction later updated this column with the current time stamp. These updates caused the row sizes to increase. However, with DB2 10 the fact that 20% of the pages were initially free meant that it took 4 hours for the free space to become consumed by the inserts and updates. Subsequently, the number of indirect references in the table began to rise. The results are shown in Figure 4-3 on page 92. Instead of measuring the performance of DB2 10, the baseline test was DB2 11 using the default value of PCTFREE_UPD and no changes to the DDL.

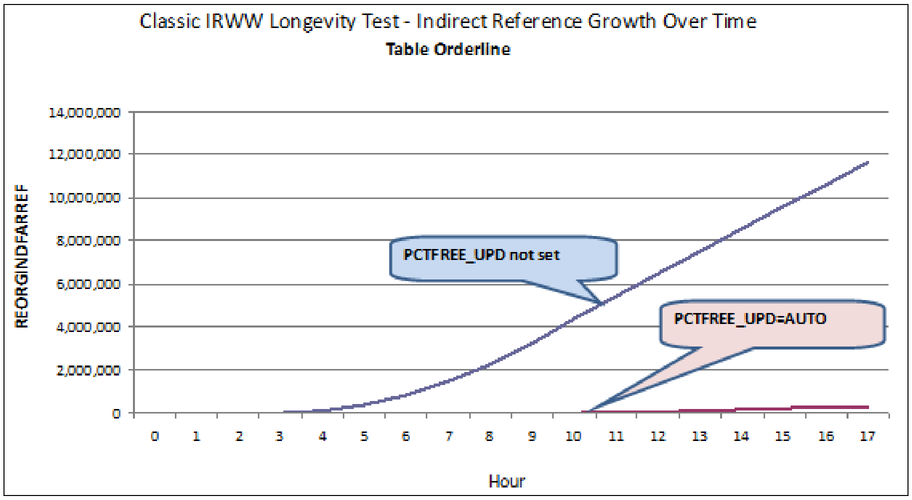

Figure 4-3 Index reference growth

After 17 hours of execution, ORDERLINE contained nearly 12 million indirect references. With DB2 11 using the PCTFREE_UPD=AUTO feature, indirect references did not begin to appear until the twelfth hour, and the number of them remained minimal. Figure 4-4 also shows that after the seventh hour of execution, PCTFREE_UPD=AUTO reduced the class 2 CPU time for the transactions that used ORDERLINE by 15 - 20%, resulting from an 80% drop in the number of getpages.

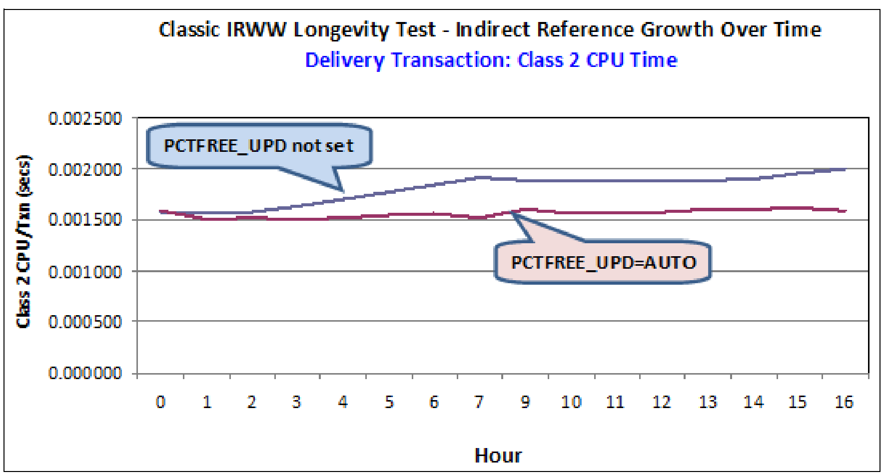

Figure 4-4 CPU time

Another table that we looked at is TORDR000, which contains a nullable CHAR(2) column. This too was compressed and beginning in the seventh hour of execution, DB2 10 began to create indirect references, as shown in Figure 4-5 on page 93. However, since the column width was so small, DB2 11 with PCTFREE_UPD=AUTO also had a difficult time avoiding indirect references. Nevertheless, by the fifteenth hour of execution, DB2 11 had reduced the number of indirect references by 20%.

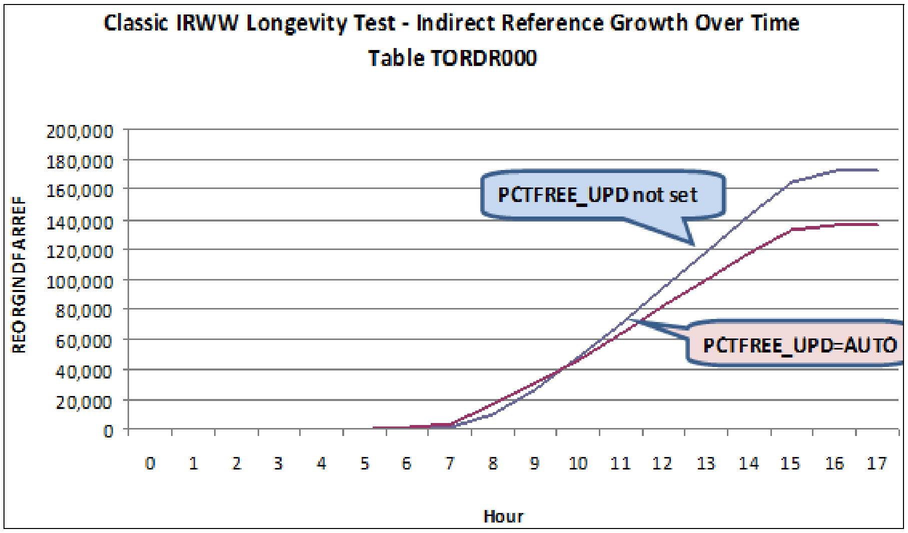

Figure 4-5 Indirect reference growth for Orders

4.3.3 Query performance

The negative impacts that are caused by indirect references on query performance are degradation by additional getpages, additional I/Os to the overflow pages, and lower clustering.

The query that we considered is random selects. Figure 4-6 shows how you can predict the effect of indirect references on the class 2 CPU time of your own random selects as a function of the buffer hit ratio and the percentage of your rows that contain an indirect reference.

For example, you observe a 50% buffer hit ratio and 10% of your rows have indirect references. The chart shows that the indirect references are causing a 4.7% increase in class 2 CPU time. Therefore, this is the amount of CPU time that you could save if DB2 avoided the indirect references. You could achieve this savings by running REORG, or you might avoid this cost starting to use DB2 11 with PCTFREE_UPD=AUTO.

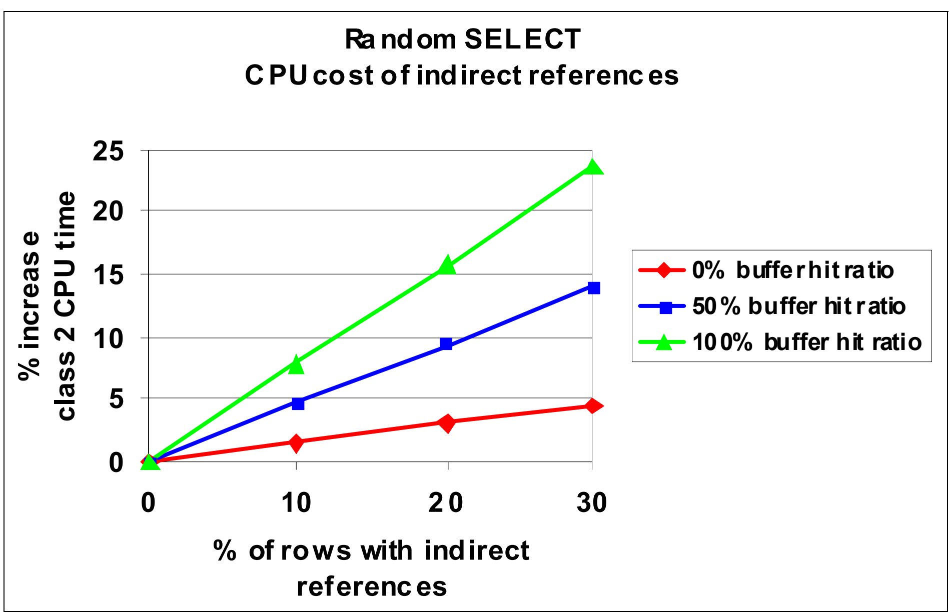

Figure 4-6 Cost of indirect references for random SELECT

The measurements used to develop the chart in Figure 4-6 on page 93 was based on a three-level index with a 100% buffer hit ratio for the index. More index levels and index sync I/Os would tend to dilute the effect of indirect references.

You can also predict the effect of indirect references on the elapsed time by making the assumption that the buffer hit ratio and sync I/O response time for the overflows is the same as it is for the pointer records. For example, if 10% of your Selects read a row that has an indirect reference, you would expect that the overflows are adding 10% more sync I/Os, with a corresponding 10% increase in the total I/O time for your Selects. Again, index sync I/Os dilute the effect of indirect references.

The effect of indirect references on a RID list scan is much greater than it is for random selects and predicting the effect on RID list scans is also much easier because 100% of the sync I/Os to the data during a RID list scan are associated with overflow records.

Lab measurements showed that if 72% of the rows contain indirect references, the indirect references increase the elapsed time of the query by 200% to 400% depending on the cache hit ratio in the storage controller.

Indirect references also impact utilities, as well as SQL table scans, but such workloads were not evaluated with indirect references.

5

IBM DB2 Analytics Accelerator

IBM DB2 Analytics Accelerator for z/OS V4.1 is the third version of the IBM Netezza® appliance solution. It brings extremely fast performance to data-intensive and complex DB2 queries in DB2 10 and DB2 11. The solution is designed to deliver the business decision as quickly as possible. This chapter describes the performance results from DB2 Analytics Accelerator V4 and DB2 11.

This chapter contains the following sections:

- Introduction to IBM DB2 Analytics Accelerator V4.1
- DB2 Analytics Accelerator V4.1 features influencing performance and availability
- Large result set improvement
- Trace enhancements with OMPE for Accelerator monitoring

5.1 Introduction to IBM DB2 Analytics Accelerator V4.1

IBM DB2 Analytics Accelerator for z/OS is a high-performance appliance that integrates Netezza technology with System z hardware and DB2 for z/OS to deliver extremely fast performance to data-intensive and complex DB2 queries for data warehousing, business intelligence, and analytic workloads. The queries requested under DB2 for z/OS can be executed orders of magnitude faster than was previously possible. The performance and cost of the Accelerator open unprecedented opportunities for organizations to use their data on the zEnterprise platform.

5.1.1 What is new in Version 4.1

DB2 Analytics Accelerator V4.1 is enhanced to support a broader set of applications, making acceleration accessible to most of DB2 for z/OS customers. V4.1 extends the available uses of the solution to include more data, queries, systems, and applications to further enhance business benefits and help realize additional cost savings with the solution.

Static SQL support

Enable acceleration of static queries as many DB2 for z/OS customers rely heavily on static SQL. This enables customers to route more queries, from a wider base of applications, to the Accelerator. In addition, the support can eliminate a great deal of the administrative task of identifying tables that have both dynamic and static SQL queries directed at them.

Support for multi-row FETCH

DB2 Analytics Accelerator V4.1 adds support for multi-row FETCH to enable the acceleration of such queries to reduce the CPU usage of processing large result sets from the Accelerator back into DB2.

Support multiple encoding

Customers who use both traditional EBCDIC content and International Unicode content in the same DB2 subsystem can now benefit from the Accelerator as V4.1 introduces support for both EBCDIC and Unicode in the same DB2 system and Accelerator.

Workload balancing and failover support

When multiple Accelerators are configured, DB2 Analytics Accelerator V4.1 can perform automated routing, based on workload balancing information, to distribute evenly the workload across the attached Accelerators. V4.1 also delivers failover support. If one Accelerator, for any reason, is unavailable, queries can be automatically rerouted to the available Accelerators.

Increased number of replication sources per Accelerator

With DB2 Analytics Accelerator V4.1, up to 10 different DB2 subsystems can be replicated into a shared Accelerator. DB2 Analytics Accelerator V3.1 can only effectively support two DB2 subsystems.

Better performance for incremental update

Incremental update in DB2 Analytics Accelerator V4.1 supports the instrumentation facility interface (IFI) filtering provided by DB2 11, which allows CPU reduction by filtering down DB2 log records to the necessary ones. This support is retrofitted to DB2 10 as well. In addition, Change Data Capture has been updated to improve apply throughput with DB2 Analytics Accelerator V4.1.

Support table load while replication is running

In IBM DB2 Analytics Accelerator V3, it is necessary to stop replication for all tables of a DB2 subsystem if one replication enabled table needs to be loaded or reloaded. With IBM DB2 Analytics Accelerator V4, loading of a replication-enabled table can be done while replication of other table continues.

Enhanced system monitoring

DB2 Analytics Accelerator V4.1 provides expanded information for monitoring applications with the IFI. The separately available IBM Tivoli® OMEGAMON XE for DB2 Performance Expert on z/OS can use this broader information to better inform customers about the usage, execution, and concurrency of queries as they are processed in the Accelerator.

High Performance Storage Saver improvements

DB2 Analytics Accelerator V4.1 automates the process that is required to make a change to archived partitions through a stored procedure. In addition, it is now possible to restore individual partitions without requiring any outages. This new version also enforces a persistent read-only state on partitions that are used by High Performance Storage Saver (HPSS). Furthermore, backup image maintenance can be simplified by marking HPSS partitions as “undeletable”.

Simplified maintenance

V4.1 offers an automated NZKit Install that facilitates the fully automatic installation and activation of Netezza kits, with no firmware or operating system updates required. It enables the IBM NPS® kit installation/activation through the same mechanisms as the Accelerator host software and enables customers to perform these maintenance tasks by themselves with the GUI or simple stored procedure calls.

5.2 DB2 Analytics Accelerator V4.1 features influencing performance and availability

We focus on features that are provided by IBM DB2 Analytics Accelerator V4.1, which are targeted to improve performance and availability:

- Offload local queries with ROWSET or multi-row fetch
- Static SQL support
- Accelerator and WLM integration
- Change Data Capture improvement
- Concurrency with incremental update

5.2.1 Offload local queries with ROWSET or multi-row fetch

Before V4, queries are not offloaded if rowset cursor or multi-row fetch (MRF) is used. V4 supports offloading both dynamic and static local queries with rowset cursor or MRF, except for SQLPL. Both DB2 10 and DB2 11 support this feature. DB2 10 support is added by APARs PM93786/PM93787/PM93788/PM93789.

The purpose of this feature is to use MRF to improve performance for an offloaded query with large result set. Using MRF can save up to 90% of z/OS side CPU time, which was spent mostly in retrieving data from DB2 Analytics Accelerator and returning data to the application. It may also improve end-to-end elapsed time if DB2 retrieving data is the bottleneck. When application fetches data very efficiently, we see queries with a large result set saved up to 80% of the end-to-end elapsed time after using MRF.

The default is to fetch 100 rows per fetch.

To benefit from this feature, changing to the application is needed if ROWSET or multi-row fetch is not yet enabled.

The following restrictions apply:

- The query acceleration of ROWSET queries or MRF does not apply to remote queries.
- For FETCH statement used for the accelerated ROWSET query:
  - All FETCH requests must be rowset fetches
  - All FETCH requests must specify a FOR N ROWS clause
  - All FETCH requests must specify the same rowset size
  - All FETCH requests must specify target host variables

Multi-row fetch measurements

Measurements are done using a set of queries executed in the Accelerator returning from a few rows to many rows. DB2 11 running on a z/OS LPAR with six dedicated zEC12 CPU, and DB2 Analytics Accelerator V4.1 on Netezza Twinfin 12 are used for the measurements. The measurements use a static application returning millions of rows through TSO batch. A default of 100 rows per fetch is used.

Figure 5-1 shows three queries using multi-row fetch and single-row fetch. All the queries with large result sets show significant amount of CPU saving. Up to 91%. This CPU time saving is CPU time on the DB2 for z/OS side.

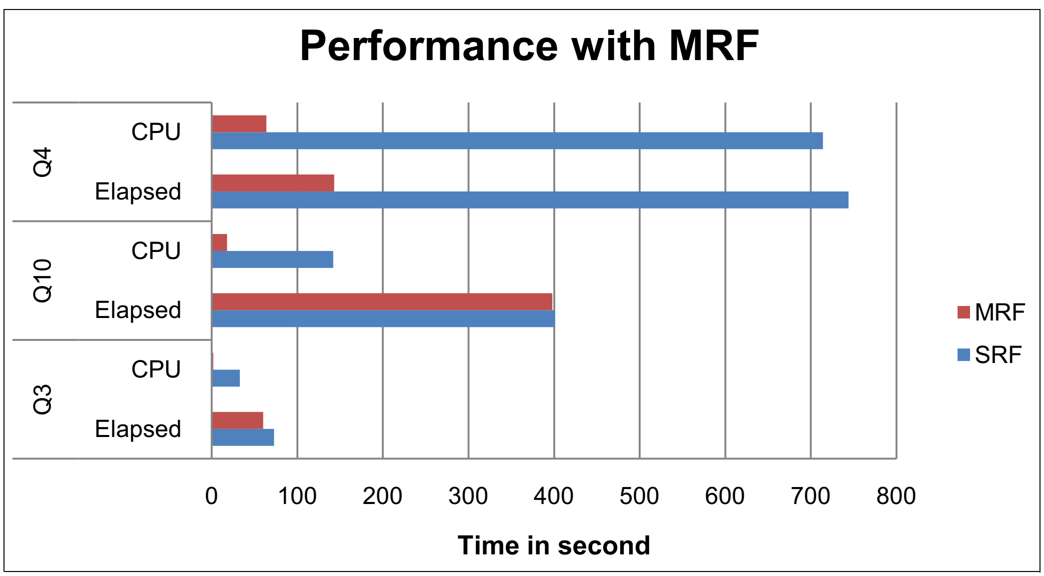

Figure 5-1 Multi-row fetch measurements

The elapsed time saving varies depending on the characteristics of queries. Q4 is very simple queries and shows 91% CPU saving as well as 80% of elapsed time reduction by using multi-row fetch, while Q10 shows 91% CPU saving but equivalent elapsed time. This is because the bottleneck of Q4 in single row fetch is in the result set processing that multi-row fetch is addressed. However, the bottleneck of Q10 is not in the result set processing. It is actual query processing in the Accelerator. Thus, improving the result set processing by multi-row fetch does not result in the elapsed time improvement.

5.2.2 Static SQL support

Before DB2 Analytics Accelerator V4.1, static SQL was not eligible to execute on the Accelerator. This restriction could impact the benefit of DB2 Analytics Accelerator for customers who have many static queries. V4.1 supports static SQL query acceleration the same way as dynamic SQL query acceleration, which means offloading only cursor queries and the SELECT portion of the INSERT FROM SELECT statements.

The static SQL SELECT INTO statements are not eligible for query acceleration.

To use this feature, the options that are shown in Example 5-1 are introduced for BIND/REBIND PACKAGE or CREATE/ALTER PROCEDURE/FUNCTION for SQLPL.

Example 5-1 New BIND/REBIND options

```
QUERY ACCELERATION  NONE | ENABLE | ENABLE WITH FAILBACK | ELIGIBLE | ALL
GET ACCEL ARCHIVE  NO | YES
```

These new bind options are not supported for CREATE/ALTER TRIGGER or REBIND TRIGGER PACKAGE.

QUERYACCELERATION BIND options

NONE No static SQL query in the application is bound for acceleration nor is accelerated

ENABLE A static SQL query is bound for acceleration if it satisfies the acceleration criteria, including the cost and heuristics criteria. If the static query does not satisfy the acceleration criteria, it is bound for execution in DB2 for z/OS. If an error condition occurs while executing the accelerated static query, DB2 fails the query and returns a negative SQL code.

ENABLE WITH FAILBACK

Same as ENABLE behavior except for error condition. If an error condition occurs, DB2 fails back the query, which means, DB2 performs a statement level incremental bind and reruns the query in DB2. After a successful OPEN is done for the query on the accelerator, failback cannot be supported.

ELIGIBLE A static SQL query is bound for acceleration if the query meets the basic acceleration criteria regardless of cost and heuristics criteria.

ALL All static SQL queries in the application are to be bound for acceleration and routed to the accelerator. If DB2 determines that a static query cannot be bound to run on the accelerator and the query references a user base table or view, DB2 fails the BIND/REBIND. A failure exception is made for declared global temporary tables (DGTTs) and created global temporary tables (CGTTs) because these tables cannot be accelerated.

GETACCELARCHIVE BIND options

NO No static SQL query is bound to retrieve archived data from the accelerator. If the static query is bound for acceleration through QUERYACCELERATION option, the query is routed to the accelerator but the query only retrieves “active” data on the accelerator but not archived data on the accelerator.

YES If bind option QUERYACCELERATION is also specified and the static query references an accelerated table that has partitioned data archived on the accelerator, and the query satisfies the criteria as specified by QUERYACCELERATION option, the query is bound acceleration, and the query retrieves both active and archived data on the accelerator at the execution.

Default values for QUERY ACCELERATION and GETACCELARCHIVE option and system parameters

The default value for both new bind options is *option not specified*, which means the user requires to specify value to be effective. The corresponding DB2 system parameters QUERY_ACCELERATION and GET_ACCEL_ARCHIVE define dynamic queries when no values are explicitly set for the CURRENT QUERY ACCELERATION special register. These system parameters do not dictate the default for static query acceleration.

For static query acceleration, system parameter ACCEL must be set either AUTO or COMMAND.

QUERY_ACCEL_OPTIONS option#2 needs to be set to 'YES' to enable the acceleration of static INSERT from SELECT statement. This is the same as for dynamic statements.

New DB2 table SYSACCEL.SYSACCELERATEDPACKAGES

For a successful bind using the new accelerated bind options, the new DB2 table SYSACCEL.SYSACCELERATEDPACKAGES needs to be created using the DB2 supplied DDL shipped with APARs PM95612 (DB2 11) and PM93789 (DB2 10). The new bind option values are stored in this table instead of SYSIBM.SYSPACKAGE.

Measurement results

Static SQL acceleration achieves similar performance as dynamic SQL acceleration. There is no noticeable performance impact to the static packages that do not use acceleration options.

Figure 5-2 shows the elapsed time of accelerated queries from selective queries in data warehousing with 5 TB database size. The measurement environment is the same as described in 5.2.1, “Offload local queries with ROWSET or multi-row fetch” on page 97.

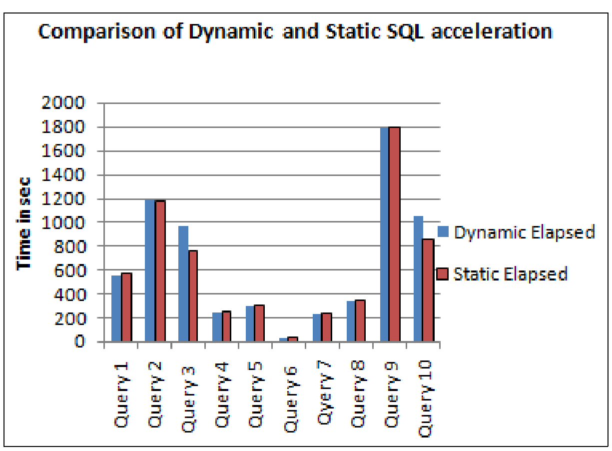

Figure 5-2 Comparing dynamic and static acceleration

Dynamic and static perform equivalently when they are accelerated, thus, the acceleration rate in static queries when it is compared to DB2 execution also stays same as dynamic:

- Data warehousing static queries show 10 - 30x speed up, on average 400x CPU reduction
- Static inserts with subselect show equivalent reduction as dynamic
- Static queries with HPSS show equivalent performance as dynamic

DSNTEP2 was used for dynamic queries and PL/I programs for static queries. The PL/I program only fetches data. There is no processing of the fetched data and no error checking, so the DB2 calls are faster in PL/I than DSNTEP2. Query 3 and 10 have large result sets, so the overall elapsed time is shorter in the static case due to simpler PL/I programs.

5.2.3 Accelerator and WLM integration

Workload management of Accelerator workloads is designed using both the z/OS workload manager (WLM) and Accelerator's workload management support. On the Accelerator, users can use guaranteed resource allocation (GRA) to partition the system so that each group receives a portion of the system resources when the group is active. These groups are called *resource sharing groups* (RSGs). GRA is enabled by default.

Each DB2 subsystem has a resource group in the Accelerator and each resource group can be defined to have a minimum guaranteed amount of resources. If a resource group is not using its entire minimum resources, those resources can be used by other resource groups. An accelerated query within a resource group has one of four priorities: Low, Normal, High, and Critical. The priority of the query is determined by the value of importance in the service class definition using z/OS WLM. All the queries compete for the resources that are allocated to that resource group.

Analytics Accelerator V4 improves the mapping of importance in z/OS WLM service class definition and priority in resource sharing group in Netezza. See Table 5-1.

Table 5-1 Mapping of importance and priority

WLM importance	Accelerator V3.1 priority	Accelerator V4.1 priority
System	Critical	Critical
Importance 1	Critical	Critical
Importance 2	High	Critical
Importance 3	Normal	High
Importance 4	Normal	Normal
Importance 5	Normal	Low
Discretionary	Low	Low

With these changes, workloads running on the Accelerator continue to use workload management support similar to z/OS.

5.2.4 Change Data Capture improvement

Incremental update is the change propagation function of the IBM DB2 Analytics Accelerator, which was made available with V3. It is based on, and is a special use of the existing Change Data Capture (CDC) component of IBM InfoSphere Data Replication. It is integrated into the

IBM DB2 Analytics Accelerator product and managed using the Accelerator Studio. It is an included feature with IBM DB2 Analytics Accelerator.

All required CDC components are included with the Accelerator:

- The capture agent is IBM InfoSphere Change Data Capture for DB2 for z/OS.
- The target agent is IBM InfoSphere Change Data Capture for IBM Netezza Linux x86.
- The management interface component is the IBM InfoSphere Change Data Capture Access Server for Linux x86.

IBM DB2 Analytics Accelerator V4.1 has added the following functional enhancements:

- Incremental update support of multiple DB2 subsystems

  The incremental update feature supports managing multiple Netezza databases by the same replication engine. Using the same replication engine allows enabling up to 10 DB2 subsystems to replicate into one shared accelerator.

- Support replication while tables are reloaded

  In IBM DB2 Analytics Accelerator V3 it was necessary to stop replication for all tables. If one replication enabled table needs to be reloaded, all replicated tables are affected. Now reloading of a table can be done while replication of other tables continues, only the reloaded table is affected.

In terms of performance, IBM DB2 Analytics Accelerator V4. 1 with DB2 11 or DB2 10 with APAR PM90568, supports the Change Data Capture Log read (IFI306) improvement, which reduces the log records read by CDC log read task.

DB2 Analytics Accelerator V4.1 performance testing configuration

Hardware configuration:

- N1001-010 (TF12)

Software configuration:

- DB2 10 with APAR PM90568 (PTF UK97013)
- ISCDC4zOS V10R2M1 with PTF UI13632

Accelerator Server: 4.1.0.201311141445

- Netezza Performance Server: 7.0.2 (P-9) [Build 33473]
- Netezza firmware: 7.0.2 (P-9) [Build 33473]
- Netezza host platform: 5.1
- Access Server: 10.2.1.2220
- Replication engine: 10.2.1 [Build CCTRJYPP_20_4]

There are seven tables that are defined with CHANGE DATA CAPTURE and COMPRESS YES. Among the seven tables, two tables are defined in the subscription to DB2 Analytics Accelerator.

Concurrent inserts and updates are run against two tables in the DB2 Analytics Accelerator subscription and single thread insert against the other five tables. The average number of inserts per second is 3117. The average number of updates per second is 924. And the average log record size is 109 bytes.

While the entire measurement runs over 20 minutes, insert into five tables finishes in 5 minutes. At peak, total insert rate from the 5-minute period reaches 8200 insert per second.

Incremental update

New DB2 statistics capture the replication latency and insert, update, and delete rows at accelerator.

In the above scenario, an average of 924 rows per second are inserted and 924 rows per second are updated at accelerator. These match with source workload insert and update rate against accelerated tables. The average latency is less than 1 minute.

At the IDAA side, an average of 1890 inserts and 925 deletes per second is reported. Since the CDC replication engine transforms one update statement in DB2 to one delete and one insert statement, this is an expected result.

IFI filtering performance improvement

As described in 2.7, “Change Data Capture Log read (IFCID 306) improvement” on page 45, IFI filtering improves performance by reducing the log records read by CDC log read task.

We focused on the CPU time from the package called CHCDSLXX, which issues the log read requests.

The specific measurement scenario described above showed 22% CPU time reduction in the CHCDSLXX package. The CPU time measurement is taken from 20-minute measurement intervals including a 5-minute period with activities against non-Accelerator table inserts. DB2 statistics reports also showed significant reduction in log records read. The latency of replication is reported within 1 minute for the 20-minute intervals.

With the addition of filtering capability, DB2 10 and DB211 can reduce the CPU utilization in DB2 without impacting replication latency.

5.2.5 Concurrency with incremental update

In Analytics Accelerator V3, the incremental update apply operations are run with normal priority and are in the same resource group as the accelerated queries. This can impact the replication performance of the incremental update in the Accelerator when higher or normal priority queries are executed concurrently.

Analytics Accelerator V4 has added support to give *preference* to query versus incremental update workload so that different workloads can get optimal Accelerator system resources to meet the performance expectation. There is a resource sharing group per subsystem and an extra dedicated group for replication is added so that the replication operation can be isolated from accelerated query operations.

The performance measurement in Figure 5-3 on page 104 shows that, with Analytics Accelerator V3, replication operations (deletes and inserts) in the Accelerator are impacted when queries with high priority come in.

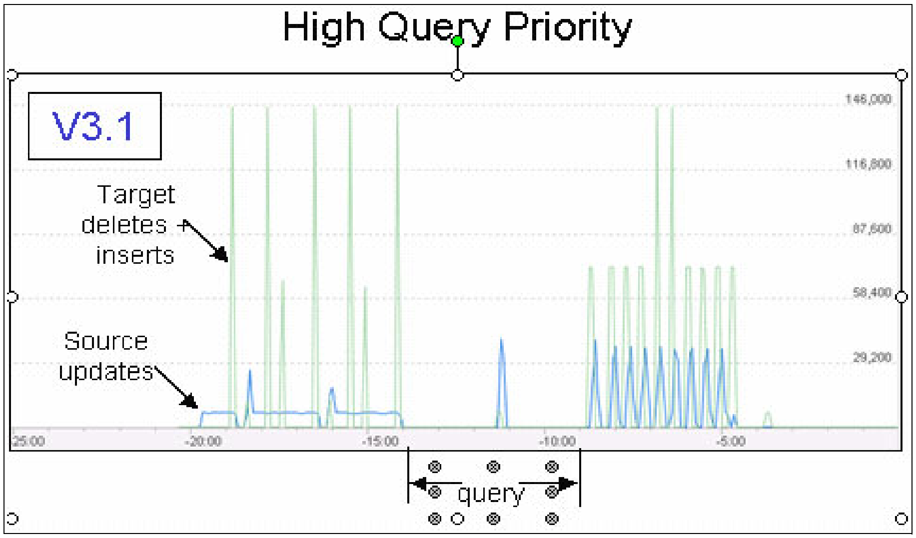

Figure 5-3 Impact of high priority query with Accelerator V3.1

Figure 5-4 shows that replication operations are executed concurrently with query processing in Analytics Accelerator V4 and demonstrates improvement in the concurrency of queries and replication operations.

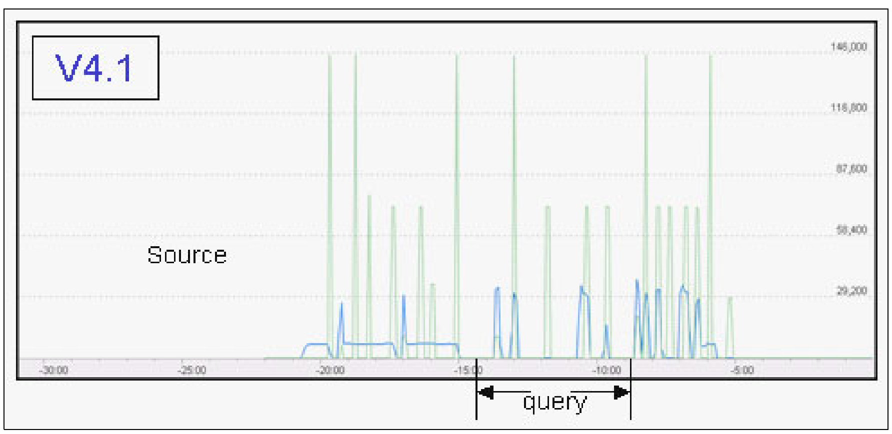

Figure 5-4 Reduced impact of high priority query with Accelerator V4.1

5.3 Large result set improvement

When queries are accelerated onto DB2 Analytics Accelerator, query processing is done outside of DB2 and DB2 receives the results back from DB2 Analytics Accelerator through IBM Distributed Relational Database Architecture™ (DRDA). Generally, DB2 CPU cost is very small when queries with a limited number of returned rows are accelerated. For example, a query returning five rows has only 4 ms DB2 CPU time when it is accelerated versus 13478s when it runs on DB2. However, when the number of returned rows is large, such as millions, DB2 may spend quite amount of CPU time receiving the rows, and sometimes it may cost more CPU to accelerate the query and receive the results than run it on DB2.

5.3.1 DB2 11 improvements on large result sets

DB2 11 reduces CPU cost in receiving DB2 Analytics Accelerator returned rows.

DB2 11 has several enhancements in DDF and run time to reduce the number of data moves and code paths. Our workload shows an average 8.4% DB2 CPU reduction in accelerated queries with large result sets.

5.3.2 Cost of returned rows from DB2 Analytics Accelerator

The table in Figure 5-5 shows the result of a lab experiment of DB2 CPU cost in microseconds per row on receiving data from DB2 Analytics Accelerator. The numbers are based on zEC12 machine with six CPUs. The numbers here are for reference. Different workloads on different DB2 environments might have different results.

Q1, Q2, Q3, Q4, and Q5 are all DB2 Analytics Accelerator accelerated queries with the same two tables joined. The difference is the type and number of selected columns:

- Q1 selects five date columns with record length of 20 bytes.
- Q2 selects five string type columns with record length of 182 bytes.
- Q3 selects five integer columns with record length of 20 bytes.
- Q4 selects 12 columns of date, string, and integer types with record length of 141 bytes.
- Q5 selects 25 columns of date, string, and integer types with record length of 275 bytes.

QUERY	#ROWS	SELECT LIST			DB2 CPU second	DB2 CPU microsec/row	Avg DB2 CPU microsec/row
		DATA TYPE	#COLS	LENGTH			
Q1	10,000	DATE	5	20	0.031	3.150	
Q1	100,000	DATE	5	20	0.276	2.757	
Q1	1,000,000	DATE	5	20	2.734	2.734	
Q1	10,000,000	DATE	5	20	27.576	2.758	2.850
Q2	10,000	CHAR/VCHAR	5	182	0.030	2.952	
Q2	100,000	CHAR/VCHAR	5	182	0.286	2.856	
Q2	1,000,000	CHAR/VCHAR	5	182	2.847	2.847	
Q2	10,000,000	CHAR/VCHAR	5	182	28.203	2.820	2.869
Q3	10,000	INT	5	20	0.029	2.922	
Q3	100,000	INT	5	20	0.278	2.782	
Q3	1,000,000	INT	5	20	2.751	2.751	
Q3	10,000,000	INT	5	20	27.606	2.761	2.804
Q4	10,000	MIXED	12	141	0.034	3.380	
Q4	100,000	MIXED	12	141	0.314	3.137	
Q4	1,000,000	MIXED	12	141	3.150	3.150	
Q4	10,000,000	MIXED	12	141	31.486	3.149	3.204
Q5	10,000	MIXED	25	275	0.037	3.686	
Q5	100,000	MIXED	25	275	0.357	3.565	
Q5	1,000,000	MIXED	25	275	3.579	3.579	
Q5	10,000,000	MIXED	25	275	35.795	3.579	3.602

Figure 5-5 CPU cost in microseconds per row

A fetch for n rows clause is used to control the number of returned rows, where n is 10,000, 100,000, 1,000,000, and 10,000,000 in the test.

The five queries have DB2 CPU time nearly proportionally aligned to the number of returned rows. The average DB2 CPU per row cost is very similar within Q1, Q2, and Q3 despite different selected data types and record length.

When the number of selected columns increases in Q4 and Q5, the average DB2 CPU per row cost increases as well, but not proportionally.

5.4 Trace enhancements with OMPE for Accelerator monitoring

IBM DB2 Analytics Accelerator for z/OS is a high-performance appliance that integrates IBM Netezza and zEnterprise technologies. The solution delivers extremely fast results for complex and data-intensive DB2 queries on data warehousing, business intelligence, and analytic workloads.

This section shows some sample batch reports from the OMPE tool for monitoring Accelerator performance:

- Accelerator modeling
- Accelerator accounting batch report enhancements
- Accelerator statistics batch report enhancements

5.4.1 Accelerator modeling

OMPE accounting report has been enhanced to provide Accelerator modeling information starting with DB2 10.

A new DB2 10 subsystem parameter ACCELMODEL and three new accounting fields have been added to provide an accelerator modeling function. Accelerator modeling allows queries to provide the CPU time and elapsed time spent processing SQL in DB2 that may be eligible for execution on an accelerator. Additionally, EXPLAIN populates DSN_QUERYINFO_TABLE, which shows if a query is eligible for offload or not.

The ACCELMODEL subsystem parameter determines whether to enable modeling of query workload for evaluating potential savings for both the accumulated elapsed time and CPU time if the plan is executed on an accelerator. Only queries that are deemed eligible for execution on an accelerator by DB2 are included in accelerator-related fields of accounting trace IFCID 3:

NO Specifies that no modeling is to be performed. This is the default setting.

YES Specifies that modeling is to be performed and considers acceleration eligibility for an SQL statement updating the new accounting fields accordingly.

To enable modeling, the Accelerator special register CURRENT QUERY ACCELERATION and subsystem parameter QUERY_ACCELERATION (set by the CURRENT QUERY ACCEL field of DSNTIP82 installation panel) must be set to NONE. All other values for the special register and subsystem parameter cause DB2 to attempt to accelerate queries instead of performing accelerator modeling.

DB2 does not perform accelerator modeling for SQL queries executed under a native stored procedure (for example, SQLPL procedure). However, DB2 does perform accelerator modeling for SQL queries executed under an external stored procedure. Accelerator modeling for native or SQLPL procedures is a consideration for the future.

EXPLAIN for a statement that goes through accelerator modeling has the following output:

(1) If the query is eligible for offload,

```
DSN_STATEMNT_TABLE.REASON = 'ACCELMODEL ELIGIBLE'
```

(2) If the query is not eligible for offload,

```
DSN_STATEMNT_TABLE.REASON = 'ACCELMODEL NOT ELIGIBLE'
```

(3) DSN_QUERYINFO_TABLE shows the accelerated or not accelerated output (same as today). Column QINAME1 has a value of 'ACCELMDL', which is normally the accelerator name.

(4) PLAN_TABLE and all other explain tables show the DB2 access path:

- Three new accounting fields are added to the QWAC of IFCID 3. These values do not include times from SQL executed in a native SQL stored procedure that are eligible for execution on an accelerator:
  - QWAC_ACCEL_ELIG_ELA is the accumulated elapsed time spent processing SQL in DB2 that may be eligible for execution on an accelerator.
  - QWAC_ACCEL_ELIG_CP is the accumulated CPU time spent processing SQL in DB2 that may be eligible for execution on an accelerator.
  - QWAC_ACCEL_ELIG_SE DS is the accumulated CPU time consumed on an IBM specialty engine while processing SQL in DB2 that may be eligible for execution on an accelerator.

Field QWP4ACMO is added to the IFCID 106 trace record to track the ACCELMODEL parameter.

Figure 5-6 on page 108 shows the DB2 10 OMPE V511 accounting report long layout for Accelerator modeling activity for potential Accelerator eligible queries.

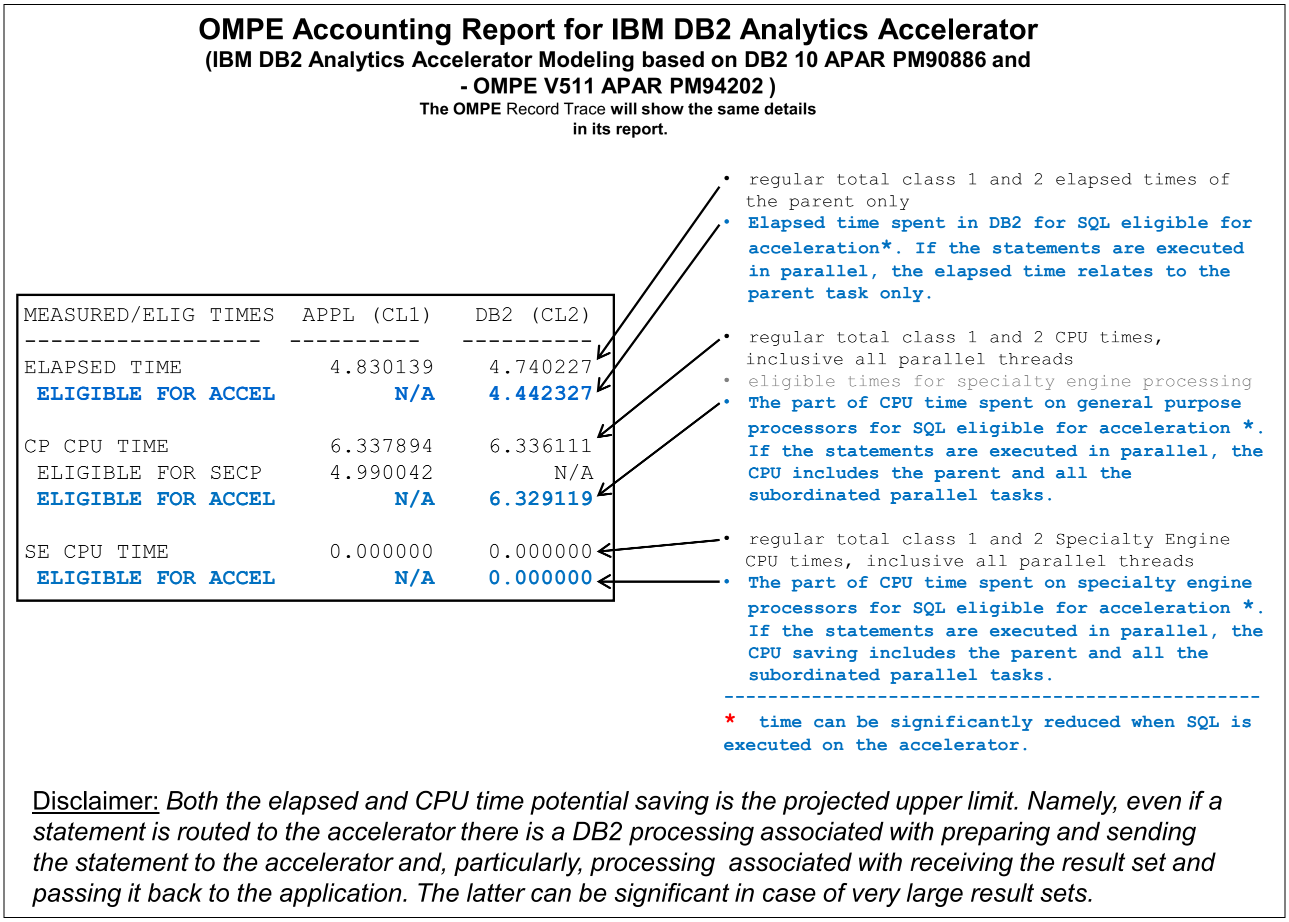

Figure 5-6 Accounting report long layout for Accelerator modeling activity

DB2 10 APAR PM90886 provides the DB2 Accelerator modeling functionality for Accelerator V3 (dynamic SQL) and DB2 10 APAR PM95035 for Accelerator V4 (static SQL).

5.4.2 Accelerator accounting batch report enhancements

The OMPE plan level accounting long report layout in Example 5-2 shows the Class 3 Accelerator (bold, blue) suspension time.

Example 5-2 Statistics long report

```
AVERAGE       APPL(CL.1)  DB2 (CL.2)  IFI (CL.5)    CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT
------------  ----------  ----------  ----------    --------------------  ------------  --------
ELAPSED TIME  6:55.50777  6:52.92867         N/P    LOCK/LATCH(DB2+IRLM)      0.000000      0.00
 NONNESTED    6:55.50777  6:52.92867         N/A     IRLM LOCK+LATCH          0.000000      0.00
 STORED PROC    0.000000    0.000000         N/A     DB2 LATCH                0.000000      0.00
 UDF            0.000000    0.000000         N/A    SYNCHRON. I/O             0.000000      0.00
 TRIGGER        0.000000    0.000000         N/A     DATABASE I/O             0.000000      0.00
                                                     LOG WRITE I/O            0.000000      0.00
CP CPU TIME    18.468298   15.586534         N/P    OTHER READ I/O            0.000000      0.00
 AGENT         18.468298   15.586534         N/A    OTHER WRTE I/O            0.000000      0.00
  NONNESTED    18.468298   15.586534         N/P    SER.TASK SWTCH            0.000032      2.00
  STORED PRC    0.000000    0.000000         N/A     UPDATE COMMIT            0.000032      2.00
  UDF           0.000000    0.000000         N/A     OPEN/CLOSE               0.000000      0.00
  TRIGGER       0.000000    0.000000         N/A     SYSLGRNG REC             0.000000      0.00
 PAR.TASKS      0.000000    0.000000         N/A     EXT/DEL/DEF              0.000000      0.00
```

```
                                                               OTHER SERVICE            0.000000      0.00
 SECP CPU        0.000000          N/A          N/A           ARC.LOG(QUIES)           0.000000      0.00
                                                              LOG READ                 0.000000      0.00
SE CPU TIME      0.000000     0.000000          N/A           DRAIN LOCK               0.000000      0.00
 NONNESTED       0.000000     0.000000          N/A           CLAIM RELEASE            0.000000      0.00
 STORED PROC     0.000000     0.000000          N/A           PAGE LATCH               0.000000      0.00
 UDF             0.000000     0.000000          N/A           NOTIFY MSGS              0.000000      0.00
 TRIGGER         0.000000     0.000000          N/A           GLOBAL CONTENTION        0.000000      0.00
                                                              COMMIT PH1 WRITE I/O     0.000000      0.00
 PAR.TASKS       0.000000     0.000000          N/A           ASYNCH CF REQUESTS       0.000000      0.00
                                                              TCP/IP LOB XML           0.000000      0.00
SUSPEND TIME     0.000000   6:36.98190          N/A           ACCELERATOR           6:36.981869    177.55
 AGENT                N/A   6:36.98190          N/A           AUTONOMOUS PROCEDURE     0.000000      0.00
 PAR.TASKS            N/A     0.000000          N/A           PQ SYNCHRONIZATION       0.000000      0.00
 STORED PROC     0.000000          N/A          N/A           TOTAL CLASS 3         6:36.981901    179.55
 UDF             0.000000          N/A          N/A

NOT ACCOUNT.          N/A     0.360236          N/A
```

The Accelerator suspension time (QWACAACW) is the average wait time for requests to the Accelerator.

The OMPE package level accounting long report layout in Example 5-3 shows the Class 3 Accelerator suspension time.

Example 5-3 Package level accounting long

```
OLAP1                VALUE                OLAP1                     TIMES    ,OLAP1,              AVERAGE TIME  AVG.EV    TIME/EVENT
------------------   ------------------   ------------------  ------------    ------------------  ------------  ------  ------------
TYPE                 PACKAGE              ELAP-CL7 TIME-AVG    2:58.529663    LOCK/LATCH              0.000000    0.00           N/C
                                          CP CPU TIME             0.000541     IRLM LOCK+LATCH        0.000000    0.00           N/C
LOCATION             TEST1                  AGENT                 0.000541     DB2 LATCH              0.000000    0.00           N/C
COLLECTION ID        PKOLAP1                PAR.TASKS             0.000000    SYNCHRONOUS I/O         0.000000    0.00           N/C
PROGRAM NAME         OLAP1                SE CPU TIME             0.000000    OTHER READ I/O          0.000000    0.00           N/C
                                          SUSPENSION-CL8       2:58.529121    OTHER WRITE I/O         0.000000    0.00           N/C
ACTIVITY TYPE        NONNESTED              AGENT              2:58.529121    SERV.TASK SWITCH        0.000028    2.00      0.000014
ACTIVITY NAME        'BLANK'                PAR.TASKS             0.000000    ARCH.LOG(QUIESCE)       0.000000    0.00           N/C
SCHEMA NAME          'BLANK'              NOT ACCOUNTED           0.000001    ARCHIVE LOG READ        0.000000    0.00           N/C
SUCC AUTH CHECK                       0   AVG.DB2 ENTRY/EXIT        210.00    DRAIN LOCK              0.000000    0.00           N/C
OCCURRENCES                           1   DB2 ENTRY/EXIT               210    CLAIM RELEASE           0.000000    0.00           N/C
NBR OF ALLOCATIONS                    1                                       PAGE LATCH              0.000000    0.00           N/C
SQL STMT - AVERAGE               103.00   CP CPU SU                  36.00    NOTIFY MESSAGES         0.000000    0.00           N/C
SQL STMT - TOTAL                    103     AGENT                    36.00    GLOBAL CONTENTION       0.000000    0.00           N/C
NBR RLUP THREADS                      1     PAR.TASKS                 0.00    TCP/IP LOB XML          0.000000    0.00           N/C
                                          SE CPU SU                   0.00    ACCELERATOR          2:58.529093    3.00     59.509698
                                                                              PQ SYNCHRONIZATION      0.000000    0.00           N/C
                                                                              TOTAL CL8 SUSPENS.   2:58.529121    5.00     35.705824
```

The package level Accelerator suspension time (QPACAACW) is the average wait time for requests to an Accelerator while executing this package.

The OMPE accounting long report layout in Example 5-4 shows the summary information for the Accelerator accounting.

Example 5-4 Accounting long report

```
ACCELERATOR  IDENTIFIER                      ACCELERATOR       AVERAGE         TOTAL  ACCELERATOR         AVERAGE         TOTAL
-----------  ----------------------------  -----------  ------------  ------------  ------------  ------------  ------------
PRODUCT      AQT04010                        OCCURRENCES          1.00            29  ELAPSED TIME
SERVER       TEST1                           CONNECTS             1.00            29   SVCS TCP/IP    6:36.980452  3:11:52.4331
                                             REQUESTS             2.00            58   ACCUM ACCEL    6:55.462728  3:20:48.4191
                                              TIMED OUT           0.00             0  CPU TIME
                                              FAILED              0.00             0   SVCS TCP/IP       0.283186      8.212407
                                             SENT                                      ACCUM ACCEL    2:02.328347   59:07.522068
                                              BYTES            3043.90         88273  WAIT TIME
                                              MESSAGES           11.00           319   ACCUM ACCEL       0.100843      2.924461
                                              BLOCKS              0.00             0
                                              ROWS                0.00             0
                                             RECEIVED
```

```
BYTES       732868175.62   21253177093
MESSAGES          185.55          5381
BLOCKS            175.55          5091
ROWS          6012167.34     174352853
```

See *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991 for the details about the fields.

5.4.3 Accelerator statistics batch report enhancements

The OMPE V520 statistics long report layout in Example 5-5 shows the Accelerator statistics summary data for the statistics interval. The two bottom sections are new for replication information.

Example 5-5 Statistics long report Summary data

```
TEST1            FOR SUBSYSTEM ONLY                QUANTITY   TEST1            TOTAL,ACCELERATOR,                 QUANTITY
-----------------------------------  --------------------   -----------------------------------  --------------------
QUERIES SUCCESSFULLY EXECUTED                         28.00   QUERIES SUCCESSFULLY EXECUTED                         28.00
QUERIES FAILED TO EXECUTE                              0.00   QUERIES FAILED TO EXECUTE                              0.00
CURRENTLY EXECUTING QUERIES                            1.01   CURRENTLY EXECUTING QUERIES                            1.01
MAXIMUM EXECUTING QUERIES                             28.00   MAXIMUM EXECUTING QUERIES                             28.00
CPU TIME EXECUTING QUERIES                  1:11:39.230000   CPU TIME EXECUTING QUERIES                  1:11:39.230000
CPU TIME LOAD/ARCHIVE/RESTORE                     0.000000   CPU TIME LOAD/ARCHIVE/RESTORE                     0.000000

CONNECTS TO ACCELERATOR                               29.00   DISK STORAGE AVAILABLE (MB)                     33557183.98
REQUESTS SENT TO ACCELERATOR                          57.00     IN USE FOR ACCEL DB - ALL DB2 (MB)             8114498.00
  TIMED OUT                                            0.00     IN USE FOR ACCEL DB - THIS DB2(MB)             3226762.00
  FAILED                                               0.00
BYTES SENT TO ACCELERATOR                          88273.00   MAXIMUM QUEUE LENGTH                                  13.00
BYTES RECEIVED FROM ACCELERATOR              21253170532.00   CURRENT QUEUE LENGTH                                   0.00
MESSAGES SENT TO ACCELERATOR                         319.00   AVG QUEUE WAIT ELAPSED TIME                        0.010229
MESSAGES RECEIVED FROM ACCEL                        5370.00   MAX QUEUE WAIT ELAPSED TIME                       38.903911
BLOCKS SENT TO ACCELERATOR                             0.00
BLOCKS RECEIVED FROM ACCELERATOR                    5090.00   WORKER NODES                                          12.00
ROWS SENT TO ACCELERATOR                               0.00   WORKER NODES DISK UTILIZATION (%)                     44.55
ROWS RECEIVED FROM ACCELERATOR                 174352753.00   WORKER NODES AVG CPU UTILIZATION (%)                  24.80
                                                              COORDINATOR CPU UTILIZATION (%)                        5.44
TCP/IP SERVICES ELAPSED TIME                3:08:53.904266   PROCESSORS                                           192.00
ELAPSED TIME IN ACCELERATOR                 3:17:49.892168   DATA SLICES                                           92.00
WAIT TIME IN ACCELERATOR                          2.828946

CPU TIME FOR REPLICATION                               N/P   CPU TIME FOR REPLICATION                          0.000000
LOG RECORDS READ                                       N/P   LOG RECORDS READ                                      0.00
  LOG RECORDS FOR ACCEL TABLES                         N/P     LOG RECORDS FOR ACCEL TABLES                        0.00
LOG RECORD BYTES PROCESSED                             N/P   LOG RECORD BYTES PROCESSED                            0.00
INSERT ROWS FOR ACCEL TABLES                           N/P   INSERT ROWS FOR ACCEL TABLES                          0.00
UPDATE ROWS FOR ACCEL TABLES                           N/P   UPDATE ROWS FOR ACCEL TABLES                          0.00
DELETE ROWS FOR ACCEL TABLES                           N/P   DELETE ROWS FOR ACCEL TABLES                          0.00
REPLICATION LATENCY                                    N/P
REPLICATION STATUS CHANGE                              N/P   ACCELERATOR SERVER START              11/19/13 19:32:40.06
                                                              ACCELERATOR STATUS CHANGE             11/19/13 19:48:19.50
```

Figure 5-7 Statistics long report Summary data

For the details about the fields, see *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991.

6

Data sharing

In this chapter, we describe the improvements to data sharing introduced with DB2 11. The main focus of these functions is availability and performance and most of them do not require user action.

This chapter contains the following sections:

- Reduction of synchronous log force writes
- Parallel castout enhancements
- Group buffer pool write around
- Improved DELETE_NAME performance
- Avoiding unnecessary U child lock propagation
- Group buffer pool write performance
- Log record sequence number spin avoidance
- Metro Mirror

6.1 Reduction of synchronous log force writes

DB2 11 reduces the number of synchronous log writes during index structure modifications in a data sharing environment to improve performance. The Index structure modifications occur during index split and index page deletion processing. This enhancement improves insert, delete, and rollback performance in a data sharing environment. Long running batch programs with a high number of inserts or deletes gain performance benefits from this enhancement.

6.1.1 Index split performance improvement

Before DB2 11, the index split logic for GBP-dependent indexes causes two synchronous log writes for each index split. DB2 11 reduces the number of log force writes from two to one with this enhancement.

In this insert scenario, the program executed about 30,000 inserts and each insert triggered an index split.

The measurements show up to 10% improvement in class 2 CPU time compared to DB2 10. See Figure 6-1.

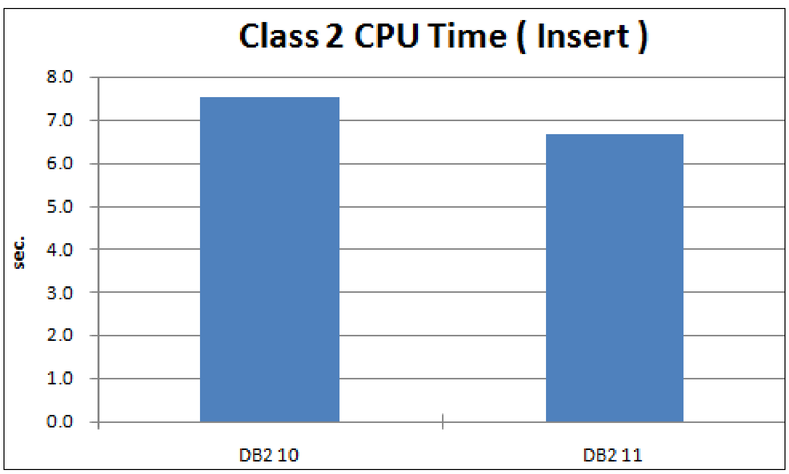

Figure 6-1 Insert: CPU time

The measurements show up to 20% improvement in class 2 elapsed time compared to DB2 10. See Figure 6-2.

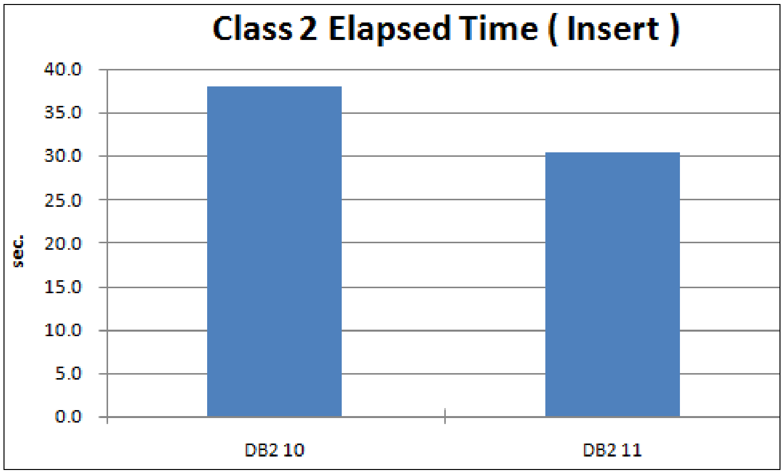

Figure 6-2 Insert: Elapsed time

6.1.2 Index page deletion performance improvement

For index page deletion for GBP-dependent indexes, DB2 10 requires five log force writes for each deletion. DB2 11 can delete index page with one to two log force writes. In this delete scenario, the program executed 800 delete statements against an index that has 32,000 leaf pages and caused 3,200 index page deletions.

Up to 19% improvements in class 2 CPU time was measured in this scenario. See Figure 6-3.

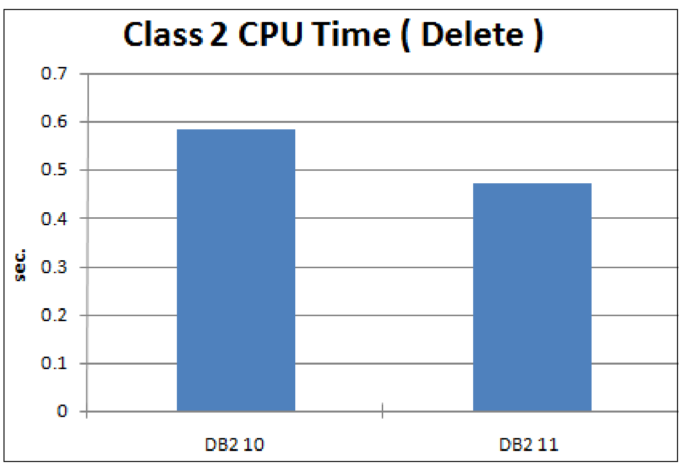

Figure 6-3 Delete: class 2 CPU time

Up to 42% improvements in class 2 elapsed time was measured in this scenario. See Figure 6-4.

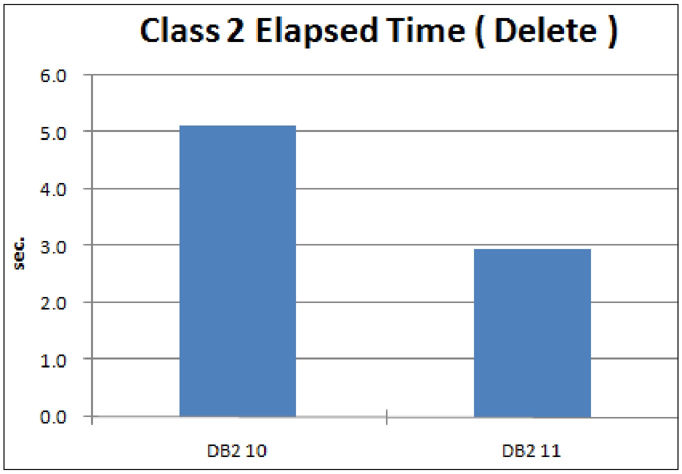

Figure 6-4 Delete: class 2 Elapsed time

6.1.3 Rollback performance improvement

Another problem related to the log force write caused by index modification is rollback performance. This happens in a scenario when there are massive deletes from an index while the index is non-GBP dependent. Then, the index becomes GBP-dependent as the rollback starts and the rollback runs for a very long time because index manager needs to force several log write I/Os when adding deleted pages back into the index tree as part of the undo of the deletes.

In this scenario, the program deleted 10,000 index leaf pages and then rolled back.

About 20% improvements in class 2 elapsed time was measured in this scenario. See Figure 6-5. Because rollback occurs in service request block (SRB) mode, there is not much reduction in class 2 CPU time.

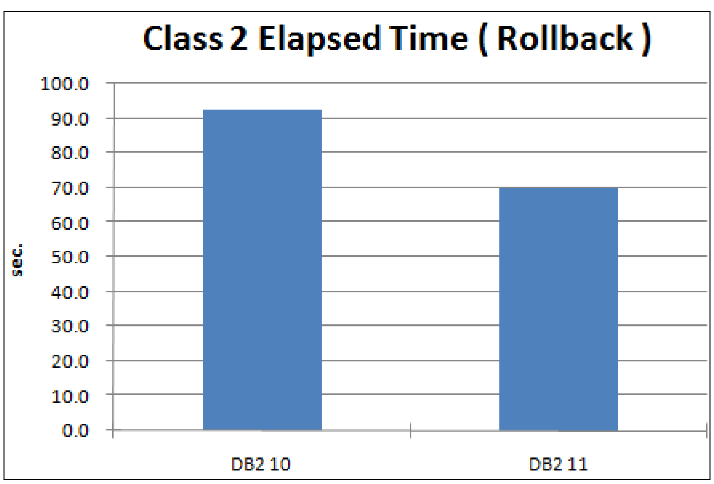

Figure 6-5 Rollback: Elapsed time

6.2 Parallel castout enhancements

Castout is the process of writing pages from the group buffer pool to disk in data sharing.

Because no physical connection exists between the group buffer pool and disk, the castout process involves reading the page from the group buffer pool into a particular member's private buffer (not part of the buffer pool storage), and then writing the page from the private buffer to disk.

6.2.1 Castout enhancements with DB2 11

One of the problems in data sharing environments with heavy write activity, before DB2 11, was that pages were written to the group buffer pools (GBPs) faster than the castout engines could process them. As a result, the group buffer pools could become congested with changed pages and, in extreme cases, group buffer pool full conditions could occur. This inefficient castout processing often resulted in application response time issues.

DB2 11 provides enhancements to make castout processing more efficient:

- Overlapped castout and write I/O resulting in reduced wait time for I/O completion
- Simplified castout notification resulting in smaller notify message size sent to castout owners
- More granular class castout threshold (CLASST castout threshold can be specified as number of pages instead of as a percentage.)

These functions are effective in DB2 11 conversion mode (CM). No user action is required to enable this function.

Both batch and online transaction processing (OLTP) workload were exercised to demonstrate the effectiveness of DB2 11 enhanced castout processing. A set of batch jobs updating a 10-partition table space were measured.

With DB2 11, the average elapsed time of the job was reduced to 19 seconds from 116 seconds in DB2 10. This is a reduction of about 84%. The reduction in elapsed time primarily comes from the reduction in the average wait time for “Update Commit” processing: from 106 seconds in DB2 10 down to 10 seconds in DB2 11. This is a reduction of 91%.

Compared to DB2 10, the GBP full condition was much relieved. In addition, the OMPE Statistics report shows a complete elimination of the “Write Failed - No Storage” and “Write to Sec-GBP Failed” conditions, indicating an efficient castout processing.

Example 6-1 shows the OMPE Statistics Report without castout improvements.

Example 6-1 OMPE Statistics Report without castout improvements

```
GROUP BP8                      QUANTITY  /SECOND  /THREAD  /COMMIT
-----------------------------  --------  -------  -------  -------
GROUP BP R/W RATIO (%)             0.00      N/A      N/A      N/A
GBP-DEPENDENT GETPAGES           249.8K   701.14   249.8K   311.87
SYN.READ(XI)-DATA RETURNED         0.00     0.00     0.00     0.00
SYN.READ(XI)-NO DATA RETURN        0.00     0.00     0.00     0.00
SYN.READ(NF)-DATA RETURNED         0.00     0.00     0.00     0.00
SYN.READ(NF)-NO DATA RETURN        0.00     0.00     0.00     0.00
UNREGISTER PAGE                    0.00     0.00     0.00     0.00

CLEAN PAGES SYNC.WRITTEN           0.00     0.00     0.00     0.00
```

```
CLEAN PAGES ASYNC.WRTN              0.00     0.00     0.00     0.00
REG.PAGE LIST (RPL) REQUEST         0.00     0.00     0.00     0.00
NUMBER OF PAGES RETR.FROM GBP       0.00     0.00     0.00     0.00
PAGES CASTOUT                     249.1K   699.04   249.1K   310.94
UNLOCK CASTOUT                   4055.00    11.38  4055.00     5.06

READ CASTOUT CLASS               2179.00     6.12  2179.00     2.72
READ DIRECTORY INFO                 0.00     0.00     0.00     0.00
READ STORAGE STATISTICS            45.00     0.13    45.00     0.06
REGISTER PAGE                       0.00     0.00     0.00     0.00
DELETE NAME                         0.00     0.00     0.00     0.00
ASYNCH GBP REQUESTS             62905.00   176.56    62.9K    78.53
EXPLICIT X-INVALIDATIONS            0.00     0.00     0.00     0.00
CASTOUT CLASS THRESHOLD           358.00     1.00   358.00     0.45
GROUP BP CASTOUT THRESHOLD          1.00     0.00     1.00     0.00
GBP CHECKPOINTS TRIGGERED           1.00     0.00     1.00     0.00

WRITE FAILED-NO STORAGE            63.00     0.18    63.00     0.08 QBGLWF
WRITE TO SEC-GBP FAILED            71.00     0.20    71.00     0.09 QBGL2F

DELETE NAME LIST SEC-GBP         4055.00    11.38  4055.00     5.06
DELETE NAME FROM SEC-GBP            0.00     0.00     0.00     0.00
UNLOCK CASTOUT STATS SEC-GBP        0.00     0.00     0.00     0.00
ASYNCH SEC-GBP REQUESTS             4.00     0.01     4.00     0.00
```

Example 6-2 shows the OMPE Statistics Report with castout improvements.

Example 6-2 OMPE Statistics Report with castout improvements

```
GROUP,BP8                       QUANTITY  /SECOND  /THREAD  /COMMIT
-----------------------------   --------  -------  -------  -------
GROUP BP R/W RATIO (%)              0.00      N/A      N/A      N/A
GBP-DEPENDENT GETPAGES            251.8K  1853.91   251.8K   314.34
SYN.READ(XI)-DATA RETURNED          0.00     0.00     0.00     0.00
SYN.READ(XI)-NO DATA RETURN         0.00     0.00     0.00     0.00
SYN.READ(NF)-DATA RETURNED          0.00     0.00     0.00     0.00
SYN.READ(NF)-NO DATA RETURN         0.00     0.00     0.00     0.00
UNREGISTER PAGE                     0.00     0.00     0.00     0.00

CLEAN PAGES SYNC.WRITTEN            0.00     0.00     0.00     0.00
CLEAN PAGES ASYNC.WRTN              0.00     0.00     0.00     0.00
REG.PAGE LIST (RPL) REQUEST         0.00     0.00     0.00     0.00
NUMBER OF PAGES RETR.FROM GBP       0.00     0.00     0.00     0.00
PAGES CASTOUT                     248.6K  1830.25   248.6K   310.33
UNLOCK CASTOUT                   4796.00    35.31  4796.00     5.99

READ CASTOUT CLASS               2537.00    18.68  2537.00     3.17
READ DIRECTORY INFO                 0.00     0.00     0.00     0.00
READ STORAGE STATISTICS            29.00     0.21    29.00     0.04
REGISTER PAGE                       0.00     0.00     0.00     0.00
DELETE NAME                         0.00     0.00     0.00     0.00
ASYNCH GBP REQUESTS             63967.00   470.99    64.0K    79.86
EXPLICIT X-INVALIDATIONS            0.00     0.00     0.00     0.00
CASTOUT CLASS THRESHOLD          1946.00    14.33  1946.00     2.43
GROUP BP CASTOUT THRESHOLD          0.00     0.00     0.00     0.00
GBP CHECKPOINTS TRIGGERED           0.00     0.00     0.00     0.00
```

```
WRITE FAILED-NO STORAGE           0.00     0.00     0.00     0.00 QBGLWF
WRITE TO SEC-GBP FAILED           0.00     0.00     0.00     0.00 QBGL2F

DELETE NAME LIST SEC-GBP       4796.00    35.31  4796.00     5.99
DELETE NAME FROM SEC-GBP          0.00     0.00     0.00     0.00
UNLOCK CASTOUT STATS SEC-GBP      1.00     0.01     1.00     0.00
ASYNCH SEC-GBP REQUESTS           5.00     0.04     5.00     0.01
```

IBM Relational Warehouse workload (IRWW) data sharing OLTP workload showed that total DB2 CPU time per transaction was reduced by about 2% with the castout processing improvement.

6.3 Group buffer pool write around

In DB2 data sharing, when batch jobs or utilities run against group buffer pool (GBP)-dependent objects, it can result in very heavy, sustained GBP page write activity. When this happens, the GBP begins to fill up with changed pages, which can result in application slowdowns or, in severe cases, pages being written to the logical page list (LPL) causing DB2 data outages.

DB2 11 addresses these issues by providing a capability to bypass writing pages to the GBP in certain situations and write the pages directly to DASD instead, while using the GBP to send buffer invalidate signals. This feature is referred to as *GBP write-around*.

Two thresholds are used to determine whether GBP write-around is invoked for all objects in a GBP or for a page set/partition:

- The GBP castout threshold (GBPOOLT)
- The class castout threshold (CLASST)

When the GBP castout threshold hits 50%, meaning that 50% of the GBP is occupied by changed pages, then write-around is invoked for writing pages for all objects.

When the class castout threshold hits 20%, meaning that 20% of a class castout queue is occupied by changed pages, then DB2 employs the write-around protocol for the page set/partition. The write-around process at the GBP level continues until the GBP castout threshold drops to 40%. The write-around process at the page set/partition level continues until the class castout threshold drops to 10%. These threshold values are fixed; the user cannot change them.

Not every page being written to GBP is eligible for GBP write-around. Only pages coming to GBP via deferred write processing, asynchronous write, are eligible. Index split and COMMIT use synchronous writes. Therefore, they are not eligible for write-around.

If either threshold is reached, and there is no interest by another member, the write-around protocol at the appropriate level is invoked. The processing occurs as follows:

- Changed pages are conditionally written to the GBP.
- Conditional write means that if a page is already cached in the GBP, the write is allowed to proceed, and if the page is not already cached, the write to GBP fails.
- For those pages that are failed to be written to GBP, they are written to DASD directly and the GBP is used to send buffer invalidate signals to the other members.

The benefit of GBP write-around is that DB2 automatically detects the flooding of writes to the GBP, and automatically responds by dynamically switching to the GBP write-around protocol for those objects that are causing the heaviest write activity. The other GBP-dependent objects using that GBP do not notice any difference in GBP access protocols; only the objects with the heaviest write activity are affected. After the GBP storage shortage has been relieved, DB2 resorts back to normal GBP write activity for all GBP-dependent objects.

This function is effective in DB2 11 CM. No user action is needed.

6.3.1 Performance measurements

A batch job was exercised to demonstrate the benefit of GBP write-around.

In this test, about 1.5 million rows were inserted into an empty table from DB2 member one of a data sharing group.

There is a unique index defined for this table. Then a batch job, updating a non-index key column from rows 1 - 170,000, is submitted from the second member. Thus GBP is allocated because of read/write interest between two data sharing members.

From the first member, a batch job updates a non-index key column from rows 200,000 - 370,000. We collected accounting and statistics traces for this batch job and observed the benefit of DB2 11 GBP write-around.

Because of MAXROWS=1 is defined for the table space, each updated row translates into a changed page. As a result of UPDATE SQL, 170,000 changed pages are written into the undersized GBP.

The summary of the measured data is reported in Table 6-1.

Table 6-1 Measurement data of a batch job with and without GBP write-around

	DB2 10	DB2 11	Notes
	No GBP write around	Yes GP write around	
Accounting (in sec.)			
Class 2 elapsed time	565.27	8.43	Significantly reduced elapsed time due to GBP write-around
CPU time	1.45	1.68	Slightly increased CPU is the trade-off for much reduced elapsed time
Suspend time	2.21	5.76	
Not accounted time	561.60	1.00	Significantly reduced not-accounted time due to no DB2 pause caused by GBP full
SQL DML			
Update	1	1	Same SQL
Rows	170,580	170,580	
BP12			
Getpages	171.100	171.400	
Buffer updates	170.600	170.600	MAXROWS=1 for table space
Pages written	0	172.600	Including pages in write around

	DB2 10	DB2 11	Notes
Pages written for castout			
GBP12			
GBP-dependent getpages	170.600	171.400	
Pages castout	170.6	146	Nearly no castout in write-around case
Explicit X-invalidation	0	170.300	Part of write-around operation
Write failed - No storage	513	1	DB2 barely pauses because of GBP full
Changed pages sync. written	282	242	Pages not eligible for write-around
Changed pages async. written	170.600	170.300	Pages eligible for write-around
Pages in write around	N/A	170.300	

With DB2 10, every changed page must be written into GBP synchronously or asynchronously. Since the castout process is relatively slow compared to the incoming changed pages during the test, DB2 pauses for one second whenever GBP full condition is detected causing application slow down. This is indicated by a significant amount of “Not Account Time” in accounting trace. GBP full situation is reported in statistics report GBP section as “Write Failed - No Storage” entry.

With DB2 11, these 170,000 changed pages are eligible for GBP asynchronous deferred write processing. In addition, they are new pages to GBP. Therefore, nearly every changed page is eligible to GBP write-around. This is recorded in statistics report, GBP section, “Pages in Write-Around” and “Explicit X-Invalidations”. Also note that in the local BP section, “Pages Written” includes pages being written around. The benefit of GBP write-around is to minimize GBP full condition and it was observed once during the test. Not-account time is 1 second in this case. Class 2 elapsed time is down to 8.43 seconds from 565.27 seconds (67 times lower), mainly due to minimizing GBP full conditions. The trade-off is slightly higher CPU time, from 1.45 to 1.68 seconds (+14%).

Example 6-3 shows the OMPE Statistics Report with GBP Write Around. Notice the “EXPLICIT X_INVALIDATIONS” counters with zero value in bold blue. Notice also the nonzero “WRITE FAILED -NO STORAGE” counters in bold blue.

Example 6-3 OMPE Statistics Report without GBP Write Around

```
GROUP,BP12                     QUANTITY  /SECOND  /THREAD  /COMMIT    GROUP BP12   CONTINUED       QUANTITY  /SECOND  /THREAD  /COMMIT
-----------------------------  --------  -------  -------  -------    -----------------------  --------  -------  -------  -------
GROUP BP R/W RATIO (%)             0.00      N/A      N/A      N/A    WRITE AND REGISTER           0.00     0.00     0.00     0.00
GBP SYN.READ(XI) HIT RATIO(%)       N/C      N/A      N/A      N/A    WRITE AND REGISTER MULT  16460.00    24.28    16.5K  8230.00
GBP-DEPENDENT GETPAGES           170.6K   251.67   170.6K    85.3K    CHANGED PGS SYNC.WRTN       28.00     0.04    28.00    14.00
SYN.READ(XI)-DATA RETURNED         0.00     0.00     0.00     0.00    CHANGED PGS ASYNC.WRTN     170.6K   251.63   170.6K    85.3K
SYN.READ(XI)-NO DATA RETURN        0.00     0.00     0.00     0.00    PAGES WRITE & REG MULT     170.6K   251.67   170.6K    85.3K
SYN.READ(NF)-DATA RETURNED         0.00     0.00     0.00     0.00    READ FOR CASTOUT           139.00     0.21   139.00    69.50
SYN.READ(NF)-NO DATA RETURN        5.00     0.01     5.00     2.50    READ FOR CASTOUT MULT    21854.00    32.24    21.9K    10.9K
UNREGISTER PAGE                    0.00     0.00     0.00     0.00
                                                                      PAGE P-LOCK LOCK REQ         0.00     0.00     0.00     0.00
CLEAN PAGES SYNC.WRITTEN           0.00     0.00     0.00     0.00     SPACE MAP PAGES             0.00     0.00     0.00     0.00
CLEAN PAGES ASYNC.WRTN             0.00     0.00     0.00     0.00     DATA PAGES                  0.00     0.00     0.00     0.00
REG.PAGE LIST (RPL) REQUEST     5351.00     7.89  5351.00  2675.50     INDEX LEAF PAGES            0.00     0.00     0.00     0.00
NUMBER OF PAGES RETR.FROM GBP      0.00     0.00     0.00     0.00
PAGES CASTOUT                    170.6K   251.67   170.6K    85.3K    PAGE P-LOCK UNLOCK REQ       0.00     0.00     0.00     0.00
UNLOCK CASTOUT                  3637.00     5.37  3637.00  1818.50
                                                                      PAGE P-LOCK LOCK SUSP        0.00     0.00     0.00     0.00
```

```
READ CASTOUT CLASS             2575.00     3.80  2575.00  1287.50    SPACE MAP PAGES          0.00     0.00     0.00     0.00
READ DIRECTORY INFO               0.00     0.00     0.00     0.00    DATA PAGES               0.00     0.00     0.00     0.00
READ STORAGE STATISTICS          13.00     0.02    13.00     6.50    INDEX LEAF PAGES         0.00     0.00     0.00     0.00
REGISTER PAGE                     0.00     0.00     0.00     0.00
DELETE NAME                       0.00     0.00     0.00     0.00   PAGE P-LOCK LOCK NEG      0.00     0.00     0.00     0.00
ASYNCH GBP REQUESTS           10885.00    16.06    10.9K  5442.50    SPACE MAP PAGES          0.00     0.00     0.00     0.00
EXPLICIT X-INVALIDATIONS          0.00     0.00     0.00     0.00    DATA PAGES               0.00     0.00     0.00     0.00
CASTOUT CLASS THRESHOLD        1532.00     2.26  1532.00   766.00    INDEX LEAF PAGES         0.00     0.00     0.00     0.00
GROUP BP CASTOUT THRESHOLD        0.00     0.00     0.00     0.00
GBP CHECKPOINTS TRIGGERED         0.00     0.00     0.00     0.00   PAGES IN WRITE-AROUND      N/A      N/A      N/A      N/A
WRITE FAILED-NO STORAGE         513.00     0.76   513.00   256.50

WRITE TO SEC-GBP FAILED           0.00     0.00     0.00     0.00
COMPL CHECKS SUSPENDED            0.00     0.00     0.00     0.00
DELETE NAME LIST SEC-GBP          0.00     0.00     0.00     0.00
DELETE NAME FROM SEC-GBP          0.00     0.00     0.00     0.00
UNLOCK CASTOUT STATS SEC-GBP      0.00     0.00     0.00     0.00
ASYNCH SEC-GBP REQUESTS           0.00     0.00     0.00     0.00
```

Example 6-4 shows the OMPE Statistics Report with GBP Write Around. Bold and blue are the contents of the fields QBGLEX and QBGLWA.

Example 6-4 OMPE Statistics Report with GBP Write Around

```
GROUP,BP12                    QUANTITY  /SECOND  /THREAD  /COMMIT   GROUP BP12   CONTINUED   QUANTITY  /SECOND  /THREAD  /COMMIT
----------------------------- --------  -------  -------  -------   -----------------------  --------  -------  -------  -------
GROUP BP R/W RATIO (%)            0.00      N/A      N/A      N/A   WRITE AND REGISTER           1.00     0.01     1.00     0.50
GBP SYN.READ(XI) HIT RATIO(%)      N/C      N/A      N/A      N/A   WRITE AND REGISTER MULT    169.5K  1801.23   169.5K    84.7K
GBP-DEPENDENT GETPAGES          171.4K  1821.34   171.4K    85.7K   CHANGED PGS SYNC.WRTN      242.00     2.57   242.00   121.00
SYN.READ(XI)-DATA RETURNED        0.00     0.00     0.00     0.00   CHANGED PGS ASYNC.WRTN     170.3K  1810.51   170.3K    85.2K
SYN.READ(XI)-NO DATA RETURN       0.00     0.00     0.00     0.00   PAGES WRITE & REG MULT     170.6K  1813.07   170.6K    85.3K
SYN.READ(NF)-DATA RETURNED        0.00     0.00     0.00     0.00   READ FOR CASTOUT            13.00     0.14    13.00     6.50
SYN.READ(NF)-NO DATA RETURN   23430.00   249.04    23.4K    11.7K   READ FOR CASTOUT MULT       21.00     0.22    21.00    10.50
UNREGISTER PAGE                  31.00     0.33    31.00    15.50
                                                                    PAGE P-LOCK LOCK REQ         0.00     0.00     0.00     0.00
CLEAN PAGES SYNC.WRITTEN          0.00     0.00     0.00     0.00    SPACE MAP PAGES             0.00     0.00     0.00     0.00
CLEAN PAGES ASYNC.WRTN            0.00     0.00     0.00     0.00    DATA PAGES                  0.00     0.00     0.00     0.00
REG.PAGE LIST (RPL) REQUEST    4635.00    49.27  4635.00  2317.50    INDEX LEAF PAGES            0.00     0.00     0.00     0.00
NUMBER OF PAGES RETR.FROM GBP     0.00     0.00     0.00     0.00
PAGES CASTOUT                   146.00     1.55   146.00    73.00   PAGE P-LOCK UNLOCK REQ       0.00     0.00     0.00     0.00
UNLOCK CASTOUT                    5.00     0.05     5.00     2.50
                                                                    PAGE P-LOCK LOCK SUSP        0.00     0.00     0.00     0.00
READ CASTOUT CLASS                4.00     0.04     4.00     2.00    SPACE MAP PAGES             0.00     0.00     0.00     0.00
READ DIRECTORY INFO               0.00     0.00     0.00     0.00    DATA PAGES                  0.00     0.00     0.00     0.00
READ STORAGE STATISTICS           4.00     0.04     4.00     2.00    INDEX LEAF PAGES            0.00     0.00     0.00     0.00
REGISTER PAGE                     0.00     0.00     0.00     0.00
DELETE NAME                       0.00     0.00     0.00     0.00   PAGE P-LOCK LOCK NEG         0.00     0.00     0.00     0.00
ASYNCH GBP REQUESTS            3301.00    35.09  3301.00  1650.50    SPACE MAP PAGES             0.00     0.00     0.00     0.00
EXPLICIT X-INVALIDATIONS        170.3K  1809.83   170.3K    85.1K    DATA PAGES                  0.00     0.00     0.00     0.00
CASTOUT CLASS THRESHOLD           1.00     0.01     1.00     0.50    INDEX LEAF PAGES            0.00     0.00     0.00     0.00
GROUP BP CASTOUT THRESHOLD        0.00     0.00     0.00     0.00
GBP CHECKPOINTS TRIGGERED         0.00     0.00     0.00     0.00   PAGES IN WRITE-AROUND      170.3K   170.3K   170.3K    85.1K
WRITE FAILED-NO STORAGE           1.00     0.01     1.00     0.50

WRITE TO SEC-GBP FAILED           0.00     0.00     0.00     0.00
COMPL CHECKS SUSPENDED            0.00     0.00     0.00     0.00
DELETE NAME LIST SEC-GBP          0.00     0.00     0.00     0.00
DELETE NAME FROM SEC-GBP          0.00     0.00     0.00     0.00
UNLOCK CASTOUT STATS SEC-GBP      0.00     0.00     0.00     0.00
ASYNCH SEC-GBP REQUESTS           0.00     0.00     0.00     0.00
```

The DB2 command **-DIS GBPOOL(GBPxx) MDETAIL** provides pages in write-around information in DSNB777I message, as shown in Example 6-5.

Example 6-5 DSNB777I message

```
DSNB777I  -CWB1   ASYNCHRONOUS WRITES
                    CHANGED PAGES                 = 170338
                    CLEAN PAGES                   = 0
                FAILED DUE TO LACK OF STORAGE     = 1
                WRITE-AROUND PAGES                = 170274
```

Hardware and software requirements

GBP write-around uses the *conditional write* coupling facility command. This support is provided in z/OS 1.12 and above, with Coupling Facility Control Code (CFCC) Level 17 and 18 on z196 and later hardware.

6.4 Improved DELETE_NAME performance

After all pages for a page set are cast out, and the page set becomes non-GBP dependent, DB2 then uses a cache DELETE_NAME request to delete both data and directory entries from the GBP. At the same time, cross-invalidation signals are sent to each member that has interest in the page set to indicate that there is no longer GBP-dependency for that page set.

Normally, this cross-invalidation process completes without any issue. In rare cases, there can be a high number of DELETE_NAME requests due to timeouts when sending cross-invalidation signals to data sharing members when there is a long distance between the members and the coupling facility.

DB2 11 resolves this issue by suppressing the cross-invalidation signals during the processing of DELETE_NAME requests.

All the necessary cross-invalidation signals have already been sent when the pages were previously written to the GBP. The cross-invalidations for DELETE_NAME are not needed after casting out the pages; therefore, the cross-invalidation signals can be suppressed.

This function is effective in DB2 CM. No user action is required to enable this function. However, new features in cross-system extended services for z/OS (XES) and Coupling Facility Control Code (CFCC) are required to use the DELETE_NAME performance enhancement. Following are the specific requirements:

- The GBP must be allocated in a coupling facility of CFLEVEL=17 or higher. The *suppress cross-invalidation* functionality is supported on the following combinations of hardware and CFCC levels:
  - z114 (2818) - DR93G CFCC EC N48162 CFCC Release 17 at the requisite microcode load (MCL) level
  - z196 (2817) - DR93G CFCC EC N48162 CFCC Release 17 at the requisite microcode load (MCL) level
  - zEC12 (2827) - CFCC Release 18
- The DB2 member that performs castout must be running on a System z server z114 or z196 that supports the suppress cross-invalidation functionality, or a zEC12.
- The following z/OS releases support the suppress cross-invalidation functionality:

  z/OS V1R12 and above with APAR OA38419 and OA44161 are installed

This feature has also been retrofitted to DB2 9 and DB2 10 via APAR PM67544 (plus PM82301 and PM88758.) No specific measurements with DB2 11 were done.

6.5 Avoiding unnecessary U child lock propagation

Before DB2 11, U child locks were propagated to the lock structure even if only one DB2 member holds S mode pageset/partition P-lock and the other members do not have any interest against the object. This can cause performance degradation in selective cases. DB2 11 avoids this unnecessary U child lock propagation.

It is a common practice to remove GBP dependency, using DB2 commands **-STOP/START DATABASE** or **-ACCESS DATABASE** before running batch program to avoid data sharing costs. However, these programs must pay additional costs caused by this unnecessary U lock propagation. This enhancement reduces CPU costs and lock structure usage for this type of program.

In case a cursor with FOR UPDATE clause holds U lock on a page or row and does not update any data yet, a DB2 member holds S mode pageset/partition P-lock on an object. If other similar applications exist on other DB2 members, it causes these DB2 members to hold S mode pageset/partition P-lock on the same object. In this case, U child locks are propagated to the lock structure to make sure that the requesting U lock is incompatible with the existing U lock. However, if only one DB2 member holds S mode pageset/partition P-lock on an object and no other interests from any other DB2 members, DB2 11 determines that U lock propagation is not necessary. DB2 11 suppresses this unnecessary U lock propagation while DB2 10 does this propagation.

As shown in Figure 6-6, up to 60% improvement in class 2 CPU time and 50% in IRLM CPU time are expected with this enhancement, while having no or very small amounts of updates using FOR UPDATE OF cursor. There is a small improvement if the cursor updates most of the fetched data. If the cursor updates the data, pageset/partition P-lock is upgraded to X from S. U child lock propagation is not done with X mode pageset/partition P-lock.

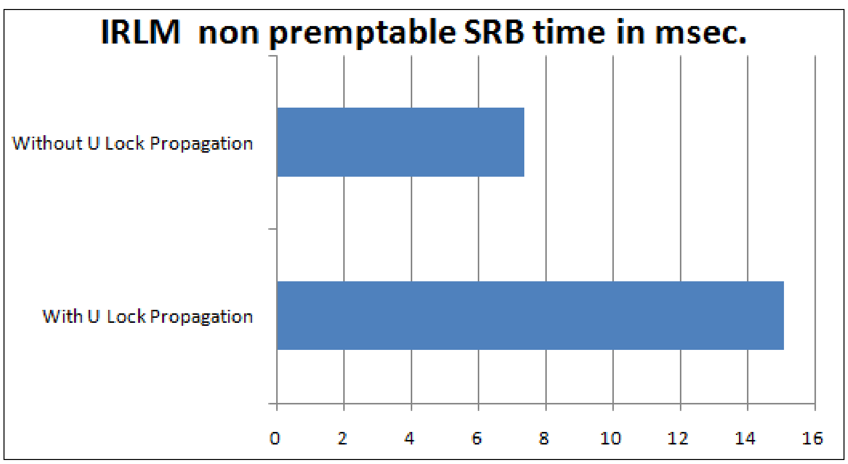

Figure 6-6 Removing U child lock propagation

6.6 Group buffer pool write performance

The GBP batch write processing in DB2 11 has been enhanced to avoid pagefix operations at the buffer level by allocating fixed storage for GBP batch write. This enhancement provides a reduction in the CPU for the COMMIT processing.

6.7 Log record sequence number spin avoidance

Enhancements were made in both DB2 9 and DB2 10 in the area of log record sequence number (LRSN) spin avoidance. DB2 9 allowed for duplicate LRSN values for consecutive log records on a given member. DB2 10 further extended LRSN spin avoidance by allowing for duplicate LRSN values for consecutive log records for inserts to the same data page.

Each of these enhancements meant that a DB2 member did not need to *spin* consuming CPU resources under the log latch to wait for the next LRSN increment. This function can avoid significant CPU overhead and log latch contention (LC19) in data sharing environments with heavy logging.

The DB2 9 and DB2 10 enhancements avoided the need to “spin” in the Log Manager to avoid duplicate LRSNs for most cases, but there were still some cases where CPU spinning was necessary, and this adds overhead. For example, consecutive DELETE or UPDATE operations to the same page required LRSN spin.

DB2 11 extends the LRSN to use more of the TOD clock precision. RBA and LRSN values have been expanded from 6 bytes to 10 bytes so it would take hundreds of years to exhaust a DB2 subsystem's or data sharing group's logging capacity based on current and projected logging rates.

Data sharing environments in DB2 11 NFM can take advantage of the larger LRSN values by avoiding LRSN spin altogether.

See 2.1, “Extended log RBA and LRSN” on page 22 for details and performance information related to this enhancement.

6.8 Metro Mirror

Metro Mirror offers a synchronous long-distance copy option that constantly updates a secondary copy of a volume to match changes made to a source volume.

Metro Mirror is needed for continuous availability (IBM HyperSwap®), protecting against any data loss at the remote site in a disaster. HyperSwap, built on top of Metro Mirror, provides continuous availability to data. Typically, you locate the second storage unit at another site some distance away from the primary site. *Synchronous mirroring* means that each update to the source unit must also be updated in the target storage unit before the host is signaled that the write has completed. This guarantees data consistency by ensuring that a write operation that completes is received by the host application after the update has been committed to the target storage unit and acknowledged by both the source and target storage units[1]. This results in near perfect data consistency but can affect write I/O response times. However, since the deferred writes and castout I/Os of DB2 are typically asynchronous regarding the application threads, the performance effects of Metro Mirror do not always have a significant effect on application response time.

But, sometimes application response time is affected. It can be very challenging to achieve good performance along with data integrity when using Metro Mirror. DB2 data sharing aggravates the problem. Alternatively, the GBP write-around feature of DB2 11 helps to relieve the problem, along with the R7.2 level of the DS8000 license code when Metro Mirror is used.

The intent of this section is to explain the complexities of the problem, how R7.2 helps to solve the problem, and how write-around also helps to relieve the problem.

Ordinarily, the extent range of a DB2 write I/O is small: at most 16 tracks in the case of non-data sharing. However, with data sharing, the castout I/Os can range across thousands

[1] Completing the writes at both storage systems before telling the host I/O is complete is normal processing, and other Metro Mirror facilities are designed to preserve data consistency in various scenarios.

of tracks. When the extent range is very large, it is a challenge for Metro Mirror to avoid deadlocks while achieving good performance. That is partly because sending N non-contiguous pages to a remote site takes far longer than sending N contiguous pages.

When Metro Mirror is not used, a specific track is serialized at the specific time that the track is written; but when Metro Mirror is used, the entire range of tracks is serialized for the entire time of the write I/O operation. Extent range contention may occur if DB2 tries to write data in parallel to the same data set extents, especially with data sharing. For example, if one DB2 castout I/O writes a few pages that are spread across 1000 tracks, a second I/O that writes any data within that range of 1000 tracks will be queued until the first I/O completes. The more non-contiguous pages that DB2 writes in the scope of one I/O, the longer it takes to replicate the data to the remote site, and the slower the castout I/O becomes. Although castout I/O is asynchronous, if the GBP fills up serious problems may occur.

The ability to do scattered writes in a single I/O is unique to z/OS, and the notion of serializing a specific track at the specific time that a track is written is unique to DB2. Since the year 2001, due to deadlock concerns, Metro Mirror was unable to bypass serialization or extent range checking at I/O startup time, and defer the serialization to the time that a specific track is written.

An alternative was to use *Global Mirror*, which asynchronously writes the pages to the remote site. Global Mirror provides data replication over extended distances between two sites for business continuity and disaster recovery. Global Mirror provides a recovery point objective of as low as 3-5 seconds between the two sites at extended distances with no performance impact on the application at the primary site. It replicates the data asynchronously and also forms a consistency group at a regular interval allowing a clean recovery of the application. Global Mirror also does not bypass serialization, but since the remote transfer is asynchronous, the length of time that an extent range is serialized is considerably less than with Metro Mirror. Global Mirror also has some performance considerations that are not present in a non-remote data replication environment, but not nearly to the same degree as Metro Mirror.

However, in 2013 IBM provided a solution for Metro Mirror environments that enables DB2 data sharing to work well. With R7.2 of the DS8870 license code, DB2's request to bypass serialization is honored in a PPRC or Metro Mirror environment. R7.2 requires a DS8870 model. Consequently, installations that have a DS8870 can now reconsider the choice of Metro Mirroring.

What about non-IBM storage or older IBM storage? You need to contact your storage vendor if it is not IBM. In any case, for older IBM storage or non-IBM storage, DB2 11 provides some relief because of DB2 castout write-around (which is explained in 6.3, “Group buffer pool write around” on page 117). As the castout I/Os slow down, the Group Buffer Pool starts to fill up, DB2 can switch to write around mode for the deferred writes. Then, the 180 page span limitation applies to the write I/Os. There is still much castout I/O because the Commits are writing to the GBP, but write around relieves some of the problem. Since castout I/O is asynchronous, if the Group Buffer Pool does not fill up, application response time is not affected by the extent conflicts. Thus, if the Commits themselves do not cause the Group Buffer Pool to fill up, application response time is good.

7

SQL

DB2 11 for z/OS provides several SQL enhancements. For the new user features and enhancements, see Chapter 8, “SQL” of *IBM DB2 11 for z/OS Technical Overview*, SG24-8180 and the DB2 manuals or the information center.

In this chapter, we discuss the SQL enhancements that relate to performance improvements. Most of them are changes in the way DB2 executes SQL statements with minimal or no user intervention needed.

This chapter contains the following topics:

- Sparse index
- Predicate indexability enhancements
- Optimizer cost adjustments
- GROUP BY and DISTINCT improvements
- DPSI enhancements for queries
- LIKE predicate evaluation enhancement
- Optimizer helping to enhance statistics
- RID pool overflow to work file
- Sort and in memory work file
- EXPLAIN enhancements
- Overriding predicate selectivities at the statement level

7.1 Sparse index

A sparse index is an in-memory structure used to efficiently access data that is accessed repeatedly and would not have efficient access using traditional index or data scanning access methods. Depending on the size of the object for which the sparse index is being built and the amount of memory allocated for sparse index support, a sparse index can range from being 100% in memory and dense, to being mostly contained in a work file with only a small sparse subset of the data being cached in memory.

Sparse indexes typically provide very efficient access to materialized query blocks, non-correlated subquery result sets, and for inner tables of joins where large amounts of data need to be scanned repeatedly and good indexing is not available. A sparse index is built by scanning an object one time and then building an optimal in memory structure which will act as either an extremely efficient 100% in-memory data cache or as some combination of an in-memory hash or index, which points to (indexes) records in a corresponding work file.

Sparse indexes for DB2 came first in with Version 4 and their use has been improved since. DB2 uses them mostly on the work files (as used by non-correlated subqueries and star joins) as a way to reduce access time and limit the impact to other users. In Explain, they are identified as PRIMARY_ACCESSTYPE = 'T' in the PLAN_TABLE when used for a join.

The MXDTCACH parameter specifies the maximum storage to be allocated for data caching per thread.

In this section, we describe the following enhancements to sparse index support:

- Hashing support for sparse index
- Sparse index performance enhancement
- Sparse index instrumentation
- Sparse index tuning

These sparse index functions are effective in DB2 11 conversion mode (CM). For static SQL, if you want to pick up new plans with new sparse index support, index duplicate skipping support, and new stage 1 predicate processing, you need to rebind. For dynamic, you need to do a full prepare.

7.1.1 Hashing support for sparse index

Sparse index support has existed since DB2 V4 for non-correlated subqueries. It was used in DB2 V7 for star joins and was opened to non-star joins in DB2 9.

DB2 11 extends the use of sparse index as an option within nested loop join by using a hashing technique. It also optimizes memory usage, including runtime validation of available system memory, and appropriately falls back to a sparse index when the result cannot be contained in memory. In DB2 11, sparse index is considered and competes based on cost just like other allowed access paths. There are some restrictions that are related to different column type or length mismatch, column expression sort, and so on.

These enhancements are available after the first REBIND for static SQL or the next dynamic SQL execution. The use of hashing for sparse indexes is controlled by subsystem parameter MXDTCACH. This subsystem parameter is conservatively set to 20 MB by default. You might consider increasing it to gain further improvement.

Although optimal setting of MXDTCACH might require some skill, the accounting and statistics reports provide a record of the number of sparse indexes where a work file was built.

This information and the number of synchronous reads in the sort work file buffer pool (see Figure 7-1) can be used as a guide to increase MXDTCACH or the sort pool.

```
BP7    READ OPERATIONS       QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------
BPOOL HIT RATIO (%)             48.36
BPOOL HIT RATIO (%) SEQU        -2.26
BPOOL HIT RATIO (%) RANDOM      97.63

GETPAGE REQUEST                618.4M    27.8K  7635.1K  7635.1K
GETPAGE REQUEST-SEQUENTIAL     305.0M    13.7K  3765.9K  3765.9K
GETPAGE REQUEST-RANDOM         313.4M    14.1K  3869.1K  3869.1K

SYNCHRONOUS READS             7419.5K   333.43    91.6K    91.6K
SYNCHRON. READS-SEQUENTIAL    6952.3K   312.44    85.8K    85.8K
SYNCHRON. READS-RANDOM         467.2K    21.00  5768.47  5768.47 <<<<
GETPAGE PER SYN.READ-RANDOM    670.74
```

Figure 7-1 Synchronous reads in the sort work file buffer pool

7.1.2 Sparse index performance enhancement

DB2 11's new sparse index support has the potential to deliver very significant CPU and elapsed time reductions for candidate SQL and workloads. Reductions of 5 - 30% in total CPU and elapsed time were measured, across nine different workloads, as a result of this new DB2 11 sparse index support. See Figure 7-2 on page 128.

Sparse index support is the single biggest contributor to the improvements of DB2 11 query performance results.

The note on w/o APREUSE in the chart is a reminder that we allowed DB2 to chose new access paths.

What type of queries will benefit?

- Queries that scan large portions of tables and queries that join tables together with little or no local filtering, especially cases where one of the tables is considerably larger than the others. Often, these queries used a sort merge join before DB2 11.
- Queries that contain materialized tables, views, or query blocks.
- Queries accessing code lookup tables, data warehousing style dimension tables, and star schemas.

Traditional sparse index support for non-correlated sub-queries and sort merge joins using CURRENT DEGREE ANY have also been enhanced in DB2 11 benefitting from the new cost model and all the related run time enhancements for DB2 11 sparse index support.

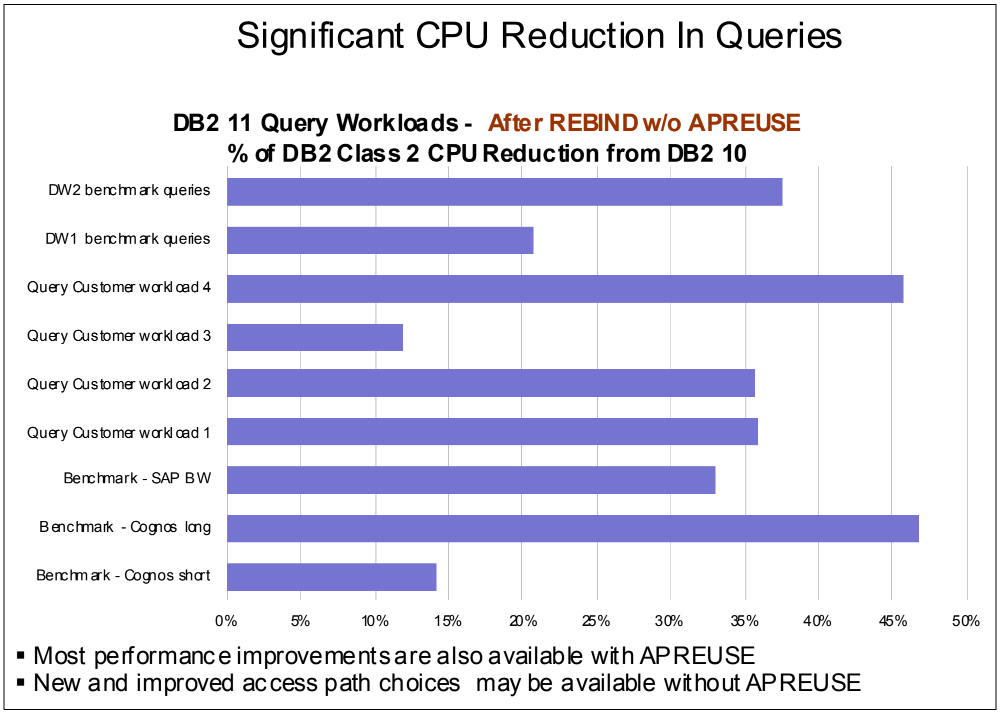

Figure 7-2 DB2 11 class 2 CPU and reductions

7.1.3 Sparse index instrumentation

PLAN_TABLE column PRIMARY_ACCESSTYPE is an easy way to identify where sparse index access is used. A PRIMARY_ACCESSTYPE value of “T” indicates that the base table or result file is materialized into a work file, and the work file is accessed via sparse index access.

New IFCID 27 fields were added to let the user know which type of sparse index is used for probing, the amount of storage used, and many other characteristics about the current sparse index. Also, new fields have been added to IFCID 2 and 3 to let the user know if the sparse index could not be optimized because not enough storage was available or if it had to be placed into a physical work file. The IFCID 27 record is written when performance trace class 3 or 9 is on. Overhead should be very small.

The IFCID 2 trace record is for the system statistics records and IFCID 3 is for accounting records. Sparse index adds two additional records to the system statistics and accounting records to let the user know the number of times that sparse index used a physical work file and the number of times that sparse index ran into storage problems where it had to disable sparse index. This information is tracked per transaction. Example 7-1 shows the instrumentation changes for IFCID 2 and IFCID 3.

Example 7-1 IFCID 2 and IFCID 3

```
QXSISTOR DS D THE NUMBER OF TIMES THAT SPARSE INDEX
WAS DISABLED BECAUSE OF INSUFFICIENT STORAGE.

QXSIWF DS D THE NUMBER OF TIMES THAT SPARSE INDEX
BUILT A PHYSICAL WORKFILE FOR PROBING.
```

7.1.4 Sparse index tuning

DB2 11's sparse index support is designed to improve performance after migration to CM with no additional tuning required.

However, as is the case with any new access path, there is always the possibility of experiencing regressions with a small subset of queries. Subsystem parameter MAX DATA CACHE (MXDTCACH) is the external knob provided to control the maximum amount of in-memory storage that DB2 allocates per SQL for sparse indexes from above the 2 GB bar pool. MXDTCACH is the amount of memory in megabytes that can be used by all the sparse indexes needed for a single query. Only as much MXDTCACH storage is allocated as is needed for any given SQL. The default value for MXDTCACH is 20 MB.

The amount of storage available for sparse index access is not limited entirely by MXDTCACH. MXDTCACH limits the in-memory portion of the sparse indexes, but sparse indexes can also overflow to work file data set storage. When this occurs, you should see field QXSIWF (SPARSE IX BUILT WF) incremented in your DB2 accounting and statistics trace data. You are also likely to see an increase in the synchronous getpage requests from your associated work file buffer pool. A sample list of the same records is in Example 7-2.

Example 7-2 Presence of SPARSE IX BUILT WF

```
MISCELLANEOUS            TOTAL
-------------------   --------
MAX STO LOB VAL (KB)         0
MAX STO XML VAL (KB)         0
ARRAY EXPANSIONS             0
SPARSE IX DISABLED           0
SPARSE IX BUILT WF           1
```

It is common place to see a nonzero value for the QXSIWF (SPARSE IX BUILT WF) counter, especially with the conservatively sized default of 20 MB for MXDTCACH, and does not necessarily indicate a performance problem. If this counter is accompanied by a large increase in work file buffer pool synchronous I/O and a corresponding increase in CPU or elapsed time, additional MXDTCACH related tuning is likely required.

While a nonzero value for QXSIWF does not guarantee a problem exists, a nonzero value for QXSISTOR (SPARSE IX DISABLED) is more likely to be indicative of a problem. This is because QXSISTOR indicates that there is insufficient system memory available to satisfy query requests for MXDTCACH. A nonzero value for QXSISTOR will most likely also result in a nonzero value for QXSIWF, but the reverse is not true. And thus addressing the nonzero QXSISTOR should be first priority.

Consider the following general sparse index tuning recommendations:

- Start with the default value of MXDTCACH of 20 MB and compare performance to your previous version of DB2 or to DB2 11 with MXDTCACH set to zero.
- If QXSISTOR (SPARSE INDEX DISABLED) is greater than zero, consider one of the following three actions as a first step to sparse index tuning:
  a. Reduce MXDTCACH, or
  b. Reduce the size of other pools (so that MXDTCACH requests succeed), or
  c. Allocate more memory to the system. It is assumed that this is the most difficult to accomplish

- If you encounter any cases of isolated regression, follow the standard recommended procedure for investigating access path performance issues. Compare the good and bad access paths and locate any additional use of primary access type “T” (sparse index).
- Ensure that proper Runstats are in place. REBIND when applicable.
- Finally, if QXSISTOR is equal to zero, you can try adjusting MXDTCACH to a larger value to see if this alleviates the regression. A simple technique is to progressively double the size of MXDTCACH and compare performance.

Total agent local storage should be monitored at the DB2 system level as you test different values of MXDTCACH in order to monitor the impact on thread storage as a result of MXDTCACH. Another area to monitor is work file buffer pool synchronous I/O and the related class 3 wait for synchronous I/O counter. It is possible, in extreme cases, to see an elapsed time regression due to a large sparse index overflow to work file resulting in inefficient synchronous I/O access to the work file data. Not all activity in the BP used for sort is sequential (especially not when sparse indexes are used). Users could also consider reducing VPSEQT by 5 - 10 percentage points lower than the current setting for work file buffer pools that experience an increase in “random” getpage access as a direct result of new sparse index access plans.

If reducing VPSEQT is a concern related to sequential sort activity, consider increasing the work file BP size by approximately the same percentage amount that VPSEQT is reduced such that the number of buffers allocated for sequential sort activity is unaffected. However, it should be noted that synchronous I/O associated with sparse index access is likely to be more costly that sequential I/O necessary to bring sort pages into the buffer pool.

7.2 Predicate indexability enhancements

Rows retrieved for a query go through two stages of processing. Certain predicates can be applied during the first stage of processing, whereas other cannot be applied until the second stage of processing. Predicates that can be applied during the first stage of processing are called *stage 1 predicates* and can be indexable. These predicates are also sometimes said to be sargable. Similarly, predicates that cannot be applied until the second stage of processing are called *stage 2 predicates*, and sometimes described as non-sargable or residual predicates.

DB2 11 is able to process some previously stage 2 predicates as stage 1 predicates allowing better performance.

In this section we look at the following topics:

- Rewrite residual predicate to indexable
- Improve indexability for CASE predicates
- Predicate pushdown
- Prune always true or false predicate
- Convert correlated subquery to non-correlated subquery
- Query rewrite to enable more choices for optimal access path selections
- Optimize invariant expressions in select list

7.2.1 Rewrite residual predicate to indexable

Before DB2 11, predicates on column expressions are generally considered as residual predicates (stage 2). DB2 cannot use an index on the column to evaluate such predicates. Index on expression (IOE) is the solution for addressing indexability issues for predicates on column expressions in DB2 9 and DB2 10. However, IOEs tend to have more overhead on insert, update, and delete performance than indexing on columns.

For the SUBSTR() function, which returns a substring of a string, an IOE is not a practical solution because users do not know the value of the length argument in advance, therefore they cannot create a proper IOE that matches the predicates.

The following predicates on YEAR(col), DATE(col), and SUBSTR(col,1,len) are simplified into sargable predicates (or stage 1) on the column directly in DB2 11:

- **`YEAR(col) op value`**

  The input column (col) is of DATE, TIMESTAMP, or TIMESTAMP WITH TIME ZONE data type. The value on the right side of the predicate is an INTEGER.

- **`DATE(col) op value`**

  The input column (col) is of TIMESTAMP data type. TIMESTAMP WITH TIME ZONE data type is supported in this case. The value on the right side of the predicate is a DATE, CHAR, or VARCHAR data type.

- **`SUBSTR(col, 1, len) op value`**

  The input column (col) is of CHAR, VARCHAR, GRAPHIC, or VARGRAPHIC data type. The data type and CCSID of the value on the right side of the predicate are the same as that of the column.

`op` represents one of the following comparison operators: **`=, >, >=, <, <=, BETWEEN`** and **`IN`**. The values on the right side of the above predicates can only be constants, variables, or invariant expressions. Invariant expressions are expressions whose results do not change during the processing of an SQL statement, as shown in Example 7-3.

Example 7-3 Invariant expression example

```
SELECT * FROM T1
   WHERE YEAR(col) = 2011;
```

If col is a date column, DB2 generates the following sargable predicate directly on col. See Example 7-4.

Example 7-4 Sample query with stage 1 predicate with date column

```
SELECT * FROM T1
   WHERE YEAR(col) = 2011
   AND  col BETWEEN '2011-01-01' AND '2011-12-31';
```

If col is a timestamp column, DB2 generates the following sargable predicate, as shown in Example 7-5.

Example 7-5 Sample query with stage 1 predicate with date column

```
SELECT * FROM T1
   WHERE YEAR(col) = 2011
   AND col BETWEEN '2011-01-01-00.00.00.0' AND '2011-12-31-24.00.00.0';
```

Example 7-6 shows a query with a SUBSTR function.

Example 7-6 SUBSTR expression in query

```
SELECT * FROM T1
WHERE SUBSTR(city, 1, 3) = 'San';
```

In this example, assume that “city” is a varchar(10) column with CCSID UNICODE: DB2 generates the sargable (stage 1) predicate shown in Example 7-7.

Example 7-7 Conversion of SUBSTR to stage 1 predicate

```
SELECT * FROM T1
   WHERE SUBSTR(city, 1, 3) = 'San'
   AND city BETWEEN X'53616E00000000000000' AND X'53616EFFFFFFFFFFFFFF';
```

In this case, X'53616E' is the hex representation of UNICODE string 'San'.

The query shown in Example 7-8 shows a SUBSTR expression compared to a host variable.

Example 7-8 SUBSTR expression compared to a host variable

```
SELECT * FROM T1
WHERE SUBSTR(city, 1, 3) = :hv;
```

In this case, DB2 generates the sargable predicate shown in Example 7-9.

Example 7-9 SUBSTR expression compared to a host variable and sargable predicate

```
SELECT * FROM T1
WHERE SUBSTR(city, 1, 3) = :hv
AND city BETWEEN  (expr) AND (expr);
```

If the value of :hv is 'San' at execution time, (expr) would use X'53616E00000000000000' and '53616EFFFFFFFFFFFFFF' for the low and the high bound values of the BETWEEN predicate.

The original predicates are still kept if and only if there is an index on expression on YEAR(col). DB2 11 makes a cost-based decision at bind time.

With this new query rewrite, queries with such predicates could show significant performance improvements. If new indexes are created to take advantage of the indexability of the predicates, access paths are changed from table space scan to index matching access. Even without new indexes created, significant performance improvement can be gained simply from changing the processing of the predicates as sargable (stage 1) instead of residual or data manager pushdown (stage 2).

7.2.2 Improve indexability for CASE predicates

CASE predicates have been another historically challenging area. DB2 11 makes progress in this area by allowing many stage 2 CASE predicates to be stage 1.

This applies to:

- Value comparison predicates whose right side contains case expressions.
- BETWEEN predicates whose low and high bound value contains case expressions.

For a value comparison predicate (**=, >, >=, <, <=, <>, LIKE, DISTINCT FROM, NOT DISTINCT FROM**):

- If its right side contains a case expression but does not contain column references or any kind of sub queries, the right side expression can be evaluated earlier and the predicate becomes sargable and indexable.

- If the case expression contains columns, it is treated as a column expression. The predicate can be turned into sargable predicate if the same conditions for column expressions are satisfied:
  – The columns in the case expression come from a table that is different from the table of the left side column of the predicate.
  – The tables referenced in the column expression appear before the left side table in join sequence.
  – The join method on the left side table should be nested loop join.

If the case expression contains any kind of subqueries, the predicate remains residual (stage 2) predicate.

Example 7-10 shows a value comparison predicate whose right side is a non-column case expression. This predicate becomes a sargable (stage 1) predicate.

Example 7-10 Value comparison predicates with right side non-column case expression

```
SELECT *
FROM T1
WHERE COL = CASE (CAST(? AS INT))
WHEN (1) THEN 'CA'
WHEN (2) THEN 'NY'
ELSE 'AL' END;
```

Example 7-11 shows a value comparison predicate whose right side is a case expression with columns. This predicate becomes a sargable predicate if the optimizer chooses nested loop join and T2 is the leading table. If there is an index on T2.C2 and an index on T1.C1, the optimizer should be able to use both indexes for the query.

Example 7-11 Value comparison predicates with right side column case expression

```
SELECT 1
FROM T1, T2
WHERE T1.C1 = CASE
WHEN (T2.INTCOL=1) THEN 'CA'
WHEN (T2.C1=2) THEN 'CA'
WHEN (T2.C1=3) THEN 'NY'
ELSE 'AL' END
AND T2.C2 =3;
```

Example 7-12 shows a BETWEEN predicate whose low operand is a non-column CASE expression. This case for local predicate can now be indexable.

Example 7-12 BETWEEN predicate with low operand non-column case expression

```
SELECT *
FROM T1
WHERE COL BETWEEN CASE (CAST(? AS INT))
WHEN (1) THEN 'CA'
WHEN (2) THEN 'NY'
ELSE 'AL' END AND 'ZZ';
```

7.2.3 Predicate pushdown

By pushing a local predicate inside a materialized view or table expression, DB2 can apply the predicates earlier and reduce the materialization cost. Generally, the more filtering the predicates to be pushed down, the more performance savings. DB2 11 continues to improve predicate push-down by removing existing restrictions.

Push down predicates on expressions

DB2 10 cannot push down the predicate in bold, blue in Example 7-13 into the materialized table expression X.

DB2 11 enables such kind of predicates to be pushed inside materialized table expressions or views. Although DB2 11 still cannot use index on expression to evaluate such predicates, it evaluates the predicates before materializing the table expressions or views, reducing the cost of materialization. Example 7-13 illustrates push down predicates on an expression with one column.

Example 7-13 DB2 push down predicates on an expression with one column

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2) AS X(C1,C2)
WHERE T1.C1 = X.C1
AND UPPER(X.C2) LIKE 'SEC%';
```

DB2 11 generates a new predicate inside the table expression as shown in Example 7-14.

Example 7-14 DB2 11 push down predicates query rewrite

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2
WHERE UPPER(T2.C2) LIKE 'SEC%') AS X(C1,C2)
WHERE T1.C1 = X.C1
AND UPPER(X.C2) LIKE 'SEC%';
```

Remove restrictions on not referenced scalar subqueries

Because replicating a scalar subquery cannot ensure that different instances of the same scalar subquery return the same result, DB2 always materializes a table expression or view that contains scalar subqueries in its outmost select list.

For the same reason, DB2 cannot push down a predicate into such view or table expressions if the predicate is on a column that maps to a scalar subquery or an expression containing subqueries in the select list of the view or table expression.

DB2 11 extends the scope of predicate pushdown if a column does not map to such an expression with subqueries.

Example 7-15 shows an example of a scalar subquery in the select list of a materialized table expression. This predicate cannot be pushed down because the column X.C2 maps to a scalar subquery inside the table expression.

Example 7-15 Scalar subquery in the select list of a materialized table expression

```
SELECT *
FROM (SELECT T2.C1, (SELECT MAX(C2) FROM T3 WHERE ...)
            FROM T2) AS X(C1, C2)
WHERE C2 = ?;
```

Example 7-16 shows a predicate that can be pushed down because the column X.C1 does not map to a scalar subquery inside the table expression.

Example 7-16 Non-scalar subquery in the select list of a materialized table expression

```
SELECT *
FROM (SELECT T2.C1, (SELECT MAX(C2) FROM T3 WHERE ...)
            FROM T2 ) AS X(C1, AS X(C1, C2)
WHERE C1 = ?;
```

Push down ON predicates with expressions

DB2 11 can push down ON predicates that satisfy the same criteria as WHERE predicates plus the following criteria:

- If a predicate is an ON predicate of a left join, the materialized view or table expression whose columns are referenced in the predicate must be the right operand of the left join.
- If a predicate is an ON predicate of an inner join, the materialized view or table expression whose columns are referenced in the predicate can be any operand of the inner join.

Example 7-17 illustrates the push down of simple ON predicates if the materialized table expression is the right side of a left join.

Example 7-17 Push down of simple ON predicates

```
SELECT *
FROM T1 LEFT JOIN
(SELECT DISTINCT T2.C1, T2.C2
FROM T2) AS X(C1,C2)
ON T1.C1 = X.C1
AND X.C2 = 'SEC';
```

For this query, DB2 generates a new predicate inside the table expression, as shown in Example 7-18.

Example 7-18 Push down of simple ON predicates query rewrite

```
SELECT *
FROM T1 LEFT JOIN
(SELECT DISTINCT T2.C1, T2.C2
FROM T2
WHERE T2.C2 = 'SEC') AS X(C1,C2)
ON T1.C1 = X.C1
AND X.C2 = 'SEC';
```

Example 7-19 shows an example of push down ON predicates of an inner join.

Example 7-19 Push down ON predicates of an inner join query

```
SELECT *
FROM (SELECT DISTINCT T2.C1, T2.C2
FROM T2) AS X(C1,C2)
INNER JOIN T1 ON T1.C1 = X.C1
AND X.C2 = 'SEC'
LEFT JOIN T3 ON X.C1 = T3.C1;
```

For this case, DB2 generates a new predicate inside the table expression, as shown in Example 7-20.

Example 7-20 DB2 11 generated predicate for inner join query

```
SELECT *
FROM (SELECT DISTINCT T2.C1, T2.C2
FROM T2
WHERE T2.C2 = 'SEC') AS X(C1,C2)
INNER JOIN T1 ON T1.C1 = X.C1
AND X.C2 = 'SEC'
LEFT JOIN T3 ON X.C1 = T3.C1;
```

Push down disjunctive predicates

Disjunctive predicates are OR predicates and their descendant predicate. They are also known as *non-Boolean Term predicates*. DB2 11 can push down disjunctive predicates into materialized table expression or view if the disjunctive predicates satisfy the following criteria:

- The top OR predicates must be a Boolean term predicate.
- All descendant predicates under the top OR operator can only reference columns from one table, which is a materialized table expression or view.
- Each descendant predicate must satisfy the push down criteria, such as the predicate cannot involve scalar fullselect, any kind of subqueries, non-deterministic UDFs, or references to any UDF parameters if the predicates are inside an SQL table UDF.

To illustrate the push down of disjunctive predicates on simple columns, consider the query in Example 7-21.

Example 7-21 SQL with disjunctive predicates on simple columns

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2) AS X(C1,C2)
WHERE T1.C1 = X.C1
AND (X.C2 BETWEEN 'SEC' AND 'SEZ'
OR X.C2 IS NULL) ;
```

DB2 generates a new predicate inside the table expression as shown in Example 7-22.

Example 7-22 Push down of disjunctive predicates on simple columns

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2
WHERE T2.C2 BETWEEN 'SEC' AND 'SEZ'
OR T2.C2 IS NULL) AS X(C1,C2)
WHERE T1.C1 = X.C1
AND (X.C2 BETWEEN 'SEC' AND 'SEZ'
OR X.C2 IS NULL) ;
```

To illustrate the push down of disjunctive predicates on column expressions, consider the query in Example 7-23.

Example 7-23 Disjunctive predicates on column expressions

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2) AS X(C1,C2)
```

```
WHERE T1.C1 = X.C1
AND (UPPER(X.C2) like 'SEC%'
OR X.C2 IS NULL);
```

DB2 generates a new predicate inside the table expression as shown in Example 7-24.

Example 7-24 Modified predicate inside table expression

```
SELECT *
FROM T1, (SELECT DISTINCT T2.C1, T2.C2
FROM T2
WHERE UPPER(T2.C2) like 'SEC%'
OR T2.C2 IS NULL) AS X(C1,C2)
WHERE T1.C1 = X.C1
AND (UPPER(X.C2) like 'SEC%' OR X.C2 IS NULL);
```

7.2.4 Prune always true or false predicate

Predicate pruning removes unneeded and pre-evaluated predicates during query transformation. Removing those types of predicates helps to improve query performance by reducing predicate evaluation overhead because the predicates are evaluated for every qualified row.

DB2 11 expands predicate pruning by removing the following restrictions:

- DB2 understands that if one branch of an AND predicate is always false, the AND predicate is always false:
  - If the AND is under another AND, DB2 marks that AND false also. Therefore, DB2 propagates the result of an always false predicate through all of its AND ancestors in the predicate tree until DB2 finds an OR predicate or the root of the predicate tree.
  - If the AND is under an OR predicate, the AND predicate and its descendants are removed. For example, **`1=3 and c1 = 1 OR c2 = 2`** is transformed into **`c2=2`**.
  - If the AND predicate is the root of the predicate tree, the subselect will be marked no row return directly.
- DB2 understands that if one branch of an OR predicate is always true, the OR predicate is always true and propagates the result up through the predicate tree:
  - If both sides of an OR are always false predicates and the OR is the root of the where predicate tree or the root of the having predicate tree, DB2 marks the subselect no row return directly and does not evaluate the subselect at run time.
  - For example, **`('%' = '%' OR C1 = 5) and C2 = :hv`** will be transformed into **`c2 = :hv`**, and **`SELECT * FROM T1 WHERE 1=4 OR 'A' = 'B';`** will be marked no row return at bind time.
- DB2 recognizes that the IS NOT NULL predicate is an always-true predicate if the column is a NOT NULL column. If the IS NULL or IS NOT NULL predicate is an always false or always true predicate, their result can be propagated through their parent AND/OR predicate and simplify a complex predicate tree.

  Example 7-25 shows a case where DB2 prunes the always true and always false predicate.

  Example 7-25 Predicates that are always evaluated to true or false are removed in DB2 11

  ```
  SELECT *
    FROM TABLE1 T1, TABLE2 T2
  WHERE (1=1 AND T1.C1 = T2.C1)
  ```

```
    OR (1=2 AND T1.C2 = T2.C2)
```

In DB2 11, predicates that are always evaluated to true would be removed[1]. Example 7-26 shows a query example with a predicate that is always evaluated to true. DB2 10 would execute the WHERE 1=1 predicate for every qualified row. DB2 11 prunes the WHERE 1=1 predicate.

Example 7-26 Predicates that are always evaluated to true are removed in DB2 11

```
SELECT *
   FROM T1
 WHERE 1=1
       AND C1 = ?;
```

7.2.5 Convert correlated subquery to non-correlated subquery

DB2 10 can replace correlated columns in a subquery with a literal if there is an equal predicate on that same column in the parent query block, and convert the correlated subquery into a non-correlated subquery if all correlated columns in that subquery are replaced. Consider the query in Example 7-27.

Example 7-27 Subquery with correlated columns

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2)
FROM T1 B
WHERE B.C1 = A.C1);
```

DB2 10 replaces A.C1 inside subquery with literal value 5. This query is rewritten as in Example 7-28.

Example 7-28 Rewritten subquery with correlated columns

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2)
FROM T1 B
WHERE B.C1 = 5);
```

This transformation enables the subquery predicate to use the index on A.C2 and can improve query performance dramatically. However, the transformation has some restrictions that are removed in DB2 11.

Support non Boolean-term subquery predicates

DB2 10 requires that the subquery predicate must be a Boolean term predicate in order to perform the query rewrite. DB2 11 removes this restriction to allow transformation from a correlated subquery to a non-correlated subquery if the subquery is on a non-Boolean term predicate. Consider the query in Example 7-29.

Example 7-29 Correlated subquery on a non-Boolean term predicate

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND (A.C2 IS NULL OR
```

[1] “OR 0=1” is NOT pruned.

```
A.C2 = (SELECT MAX(B.C2) FROM T1 B WHERE B.C1 = A.C1))
```

In this example, the subquery predicate is not a Boolean term, but the predicate A.C1=5 is a Boolean term, which must be satisfied by all the qualified rows from table A. Therefore, the correlated column A.C1 can be replaced with literal 5, and the subquery is converted into non-correlated subquery as shown in Example 7-30.

Example 7-30 Transformed correlated subquery

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND (A.C2 IS NULL OR
A.C2 = (SELECT MAX(B.C2) FROM T1 B WHERE B.C1 = 5
```

Support sub queries with UNION and UNION ALL

DB2 11 removes the restriction of not being able to convert correlated subqueries with UNION ALL to non-correlated subquery. An example is shown in Example 7-31.

Example 7-31 Correlated subqueries with UNION ALL

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2)
FROM (SELECT * FROM T1
UNION ALL
SELECT * FROM H1) AS B
WHERE B.C1 = A.C1 );
```

DB2 10 does incomplete transformation when union distribution happens: The original correlated predicate B.C1 = A.C1 is distributed into the table expression, correlated column A.C1 is duplicated, and one of them is replaced with literal 5, but the other is not. Therefore, the subquery is still correlated. Example 7-32 shows a query example.

Example 7-32 Query with union distribution in DB2 10

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2)
FROM (SELECT MAX(B.C2) FROM T1 B
WHERE B.C1 = A.C1
UNION ALL
SELECT MAX(H1.C2) FROM H1
WHERE H1.C1 = 5) AS B );
```

DB2 11 makes all correlated columns in this query replaced with literal constant, and the subquery is converted to a non-correlated subquery as shown in Example 7-33.

Example 7-33 Query with union distribution in DB2 11

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2)
FROM (SELECT MAX(B.C2) FROM T1 B
WHERE B.C1 = 5
UNION ALL
SELECT MAX(H1.C2) FROM H1
WHERE H1.C1 = 5) AS B );
```

Different data type and length

DB2 10 requires that the data type and length of the correlated column matches exactly the data type and length of the column on the other side of the equal predicate inside the subquery.

With the definition of T1.C1 SMALLINT NOT NULL and T2.C1 INT, DB2 10 cannot transform the subquery in Example 7-34 into a non-correlated subquery.

Example 7-34 Subquery example

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2) FROM T2 B WHERE B.C1 = A.C1);
```

DB2 11 can transform the correlated subquery into non-correlated subquery as shown in Example 7-35.

Example 7-35 DB2 11 transformation of correlated subquery into non-correlated subquery

```
SELECT * FROM T1 A
WHERE A.C1 = 5
AND A.C2 = (SELECT MAX(B.C2) FROM T2 B WHERE B.C1 = 5);
```

However, this enhancement does not apply if A.C1 equals host variable or parameter marker.

7.2.6 Query rewrite to enable more choices for optimal access path selections

DB2 11 extends the scope of query transformation that helps the optimizer to select a better performing access path.

Transforming non-Boolean term IN predicates into Equal and OR predicates

A Boolean term predicate is a predicate that, when it is evaluated false for a particular row, makes the entire WHERE clause false for that particular row. Boolean term predicates are preferred for single matching index access.

In other words, the OR predicate requires DB2 to evaluate the other side while AND does not. So ORs require multi-index or range-list, while ANDs can use single index access, which is generally preferred.

For many releases DB2 has rewritten the non-Boolean term OR to an IN *if the ORs are on the same column*.

For example, the following statement

```
SELECT * FROM T1 WHERE C1 = ? OR C1 = ?;
```

is rewritten as

```
SELECT * FROM T1 WHERE C1 IN (?, ?)
```

The IN-list then allows single matching index access.

More complex Boolean term predicates might be candidates for multi-index access or range list access. But although disjunctive predicates can use range list access or multi-index access plan due to the OR condition, those two access plans have many restrictions. For example, neither range list access or multi-index access is eligible for IN predicates.

Consider the following predicates:

- C1 > value OR C1 IS NULL
- C1 >= value OR C1 IS NULL
- C1 = value OR C1 IS NULL
- C1 IN (value1, value2, …) OR C1 IS NULL

All of these predicates are rewritten in DB2 11. The first three predicates in the list above are candidates for multi-index or range-list in DB2 10, but in DB2 11 they are rewritten to a predicate that is eligible for single matching index access. The first two are rewritten to a "NULL-tolerant" greater than or "greater than or equal to" but the OR is merged into the range predicate. The third is converted to an IN such as C1 IN (value, NULL). The fourth predicate, a non-matching predicate for an index on C1, is rewritten to an IN in DB2 11.

Example 7-36 shows a query with a non-Boolean term IN predicate.

This example has two IN-lists that are on different columns (one on C1, and one on C2). It cannot be rewritten into a single boolean term predicate. Because of an implementation limitation in DB2, neither range list access or multi-index access is eligible for IN list predicates. If an IN list is not a Boolean term, the IN list predicate is not indexable.

Example 7-36 Non-Boolean term IN predicate

```
SELECT * from T1
WHERE C1 IN (1,2,3)
OR C2 IN ('A', 'B', 'C');
```

DB2 11 improves the indexability of non-Boolean term IN list predicates by rewriting a non-Boolean term IN list predicate on a base table column to an equivalent OR predicate so that DB2 can choose matching using multiple indexes (one leg for each predicate). In DB2 11, the query Example 7-36 is transformed as shown in Example 7-37.

Example 7-37 DB2 11 rewrite of non-Boolean term IN predicate

```
SELECT * from T1
WHERE C1 = 1 OR C1 = 2 OR C1 = 3
OR C2 = 'A' OR C2 = 'B' OR C2 = 'C';
```

DB2 11 is likely to use a multi-index access plan instead of table space scan when there is one index with C1 as the leading column and another index with C2 as the leading column. With this access path change, performance of the query could be many times better.

Improve indexability for IN/NOT IN non-correlated subquery

IN subquery predicates are indexable already for SELECT statements. However, INSERT, UPDATE, and DELETE statements require that the data type and length of the column on the left hand side of the predicate match exactly the result date type and length of the subquery.

DB2 11 relaxes this restriction. Hence, IN subquery predicates become indexable for DELETE and UPDATE statements when the result length of the subquery is smaller than the length of the left side column.

With this enhancement, UPDATE and DELETE statements could gain much better performance with the following choices of access path that are better than table space scan.

- If there is an index on the left side column of the IN predicate, DB2 can use that index to evaluate the IN predicate.
- If there is no index on the left side column of the IN predicate, DB2 can build a sparse index on the materialized subquery and use the sparse index to evaluate the IN predicate.

Considering Customer2.c_name is char(20), custnull.c_name is char(15), Example 7-38 illustrates a query that takes advantage of this change.

Example 7-38 Improve indexability for IN/NOT IN non-correlated subquery

```
UPDATE CUSTOMER2 C1
             SET C1.C_ACCTBAL = C1.C_ACCTBAL + 100
        WHERE C1.C_NAME IN  (SELECT C2.C_NAME FROM CUSTNULL C2 WHERE C2.C_CUSTKEY < 14000);
```

Laboratory performance measurements showed up to 99% CPU time and elapsed time improvement for cases when sparse index is used, and also significant reduction of getpage counts for work file buffer pool.

7.2.7 Optimize invariant expressions in select list

Invariant expressions are expressions whose results do not change during the process of an SQL statement. Before DB2 11, if these expressions appear in select list, they are evaluated for each qualified row. DB2 11 optimized queries with invariant expressions so that those expressions are only evaluated once during the process of the whole statement. Example 7-39 illustrates a query with invariant expressions.

Example 7-39 SQL query with invariant expressions

```
SELECT CURRENT DATE + 3 DAYS,
CURRENT DATE + 4 DAYS,
CURRENT DATE + 5 DAYS,
CURRENT DATE + 6 DAYS,
CURRENT DATE + 7 DAYS,
CURRENT DATE + 8 DAYS,
CURRENT DATE + 9 DAYS,
CURRENT DATE + 10 DAYS,
CURRENT DATE + 11  DAYS,
CURRENT DATE + 12  DAYS
FROM ORDERS WHERE O_ORDERKEY < 300000;
```

Laboratory performance measurements show significant DB2 class 2 CPU time savings up to 50%. The more costly the invariant expressions are, the more the CPU time savings.

7.3 Optimizer cost adjustments

This section describes the following optimizer cost adjustments:

- Reduced sensitivity to CPU speed
- Reduced sensitivity to index NLEVELS
- Cost model recognition of new DB2 11 features

7.3.1 Reduced sensitivity to CPU speed

The DB2 11 optimizer cost model has been adjusted to be less sensitive to differences in single engine CPU speeds. There are several advantages to this new approach, with the most obvious advantages being consistency and stability of access path selections across heterogeneous environments. With DB2 11, access paths are much less likely to change due to differing CPU speeds between otherwise similar environments.

Traditional challenges encountered with access path differences when members of the same data sharing group run on System z servers with different CPU models or speeds should be much less common with the DB2 11 optimizer. Similarly, with DB2 11, it should be much easier to model a production environment's access path selections on a test environment that does not have the same CPU model as production.

7.3.2 Reduced sensitivity to index NLEVELS

DB2 11 has new cost model changes to minimize the impact of NLEVELS (the depth of an index tree) on access path selections. As a result, optimizer's choice of winning index is based predominantly on selectivity, clustering, and sort avoidance, with very little consideration given to differences in NLEVELS between competing indexes. All non-leafs are now assumed to be present in the buffer pool, so a difference in NLEVELS will have a much smaller impact than in prior releases.

7.3.3 Cost model recognition of new DB2 11 features

As with every new release of DB2, there are several new cost model improvements delivered with DB2 11. Some of these improvements are designed to make existing access path logic more robust and some are designed to support various new features in DB2 11. We highlight three of these cost model changes to support new DB2 11 features:

- The DB2 11 optimizer has been enhanced to open sparse index support for many more access paths than in previous releases. Many adjustments to the cost model have been made in order to improve our sparse index related costing to enhance this new feature.
- Cost model support has also been added to maximize the use and benefit of the new DB2 11 index duplicate skipping and early out functionality.
- The DB2 11 cost model has been improved to recognize and properly cost the correlated subquery result cache.

7.4 GROUP BY and DISTINCT improvements

This section describes:

- GROUP BY, DISTINCT, or non-correlated subqueries sort avoidance
- GROUP BY or DISTINCT early out processing in multi-table joins

The functions described in this section are effective in DB2 11 CM. No user action is necessary.

7.4.1 GROUP BY, DISTINCT, or non-correlated subqueries sort avoidance

DB2 11 provides improved run time processing for GROUP BY, DISTINCT, and non-correlated subquery sort avoidance.

Before DB2 11, queries that qualified for GROUP BY, DISTINCT, or non-correlated subquery sort avoidance would use a supporting index to ensure order, discarding duplicate entries, and avoiding a formal sort. The sort was avoided, but every index entry was still processed.

DB2 11 takes sort avoidance for GROUP BY, DISTINCT, and non-correlated subqueries one step further by only processing the first value within each set of duplicate index values and then jumping ahead to the next needed index key value. This duplicate skipping can reduce the number of index keys and record identifiers (RIDs) processed and the number of index leaf pages scanned.

Figure 7-3 illustrates a simplified example of how DB2 11 can use the non-leaf key information to skip forward to find the next distinct key value for a GROUP BY. This technique is applicable for DISTINCT, GROUP BY, and non-correlated subqueries. Before DB2 11, DB2 would scan the leaf pages and remove duplicates before returning the data to the application. DB2 11 can remove the duplicates earlier by using index look aside (leaf high key to non-leaf) and skipping them within the index, regardless of whether the distance between distinct entries is short or long. A greater performance benefit is gained when whole index leaf pages can be skipped.

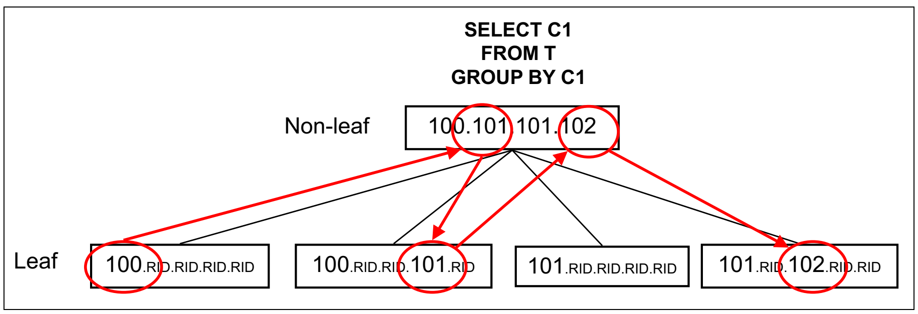

Figure 7-3 DB2 11 simplified index key skipping

7.4.2 GROUP BY or DISTINCT early out processing in multi-table joins

Optimization of DISTINCT and other duplicate removal patterns extends to join queries in DB2 11 where the join is coded as an existence check. In such queries, any duplicates that are introduced from the join are not required for the final result.

DB2 11 can select an early-out join for the inner table as soon as the first match is found rather than processing all matching rows. Before DB2 11, this type of early-out join was available only to correlated EXISTS subqueries that were transformed to a join.

In DB2 11, each inner table probe stops after the first match is found for the query that is shown in Example 7-40.

Example 7-40 Query with inner table probe

```
SELECT DISTINCT T1.*
FROM T1, T2
WHERE T1.C1 = T2.C1
```

Early-out join also applies to non-Boolean term join conditions with an early-out table, as shown in Example 7-41.

Example 7-41 Non-Boolean term join conditions

```
SELECT DISTINCT T1.*
FROM T1, T2
```

```
WHERE T1.C1 = 1
   OR T1.C1 = T2.C1
```

Correlated subqueries can also have improved performance in DB2 11 because of optimized usage of a subquery cache that has existed since DB2 V2. Example 7-42 shows a simple example of a common query pattern that is used in temporal-based or time-based implementations where the most recent (or least recent) version is required by the query. DB2 11 optimizes this pattern by recognizing when order is provided and by adjusting the cache size as needed, thus streamlining the cache lookup process.

Example 7-42 Correlated subquery pattern optimized in DB2 11

```
SELECT *
FROM POLICY P1
WHERE P1.POLICY_DATE =
(SELECT MAX(P2.POLICY_DATE)
 FROM POLICY P2
 WHERE P2.CUSTNO = P1.CUSTNO)
```

Several tests were done to measure the performance impacts of index duplicate skipping. The query used was a simple DISTINCT query as shown in Example 7-43.

Example 7-43 Benchmark query for duplicate skipping test

```
SELECT DISTINCT COL1 FROM TABLE;
```

Figure 7-4 shows the DB2 11 CPU savings compared to DB2 10 for sort avoidance queries, which can take advantage of duplicate skipping. The chart shows the CPU reduction in % in relation to the number of RIDs per index key.

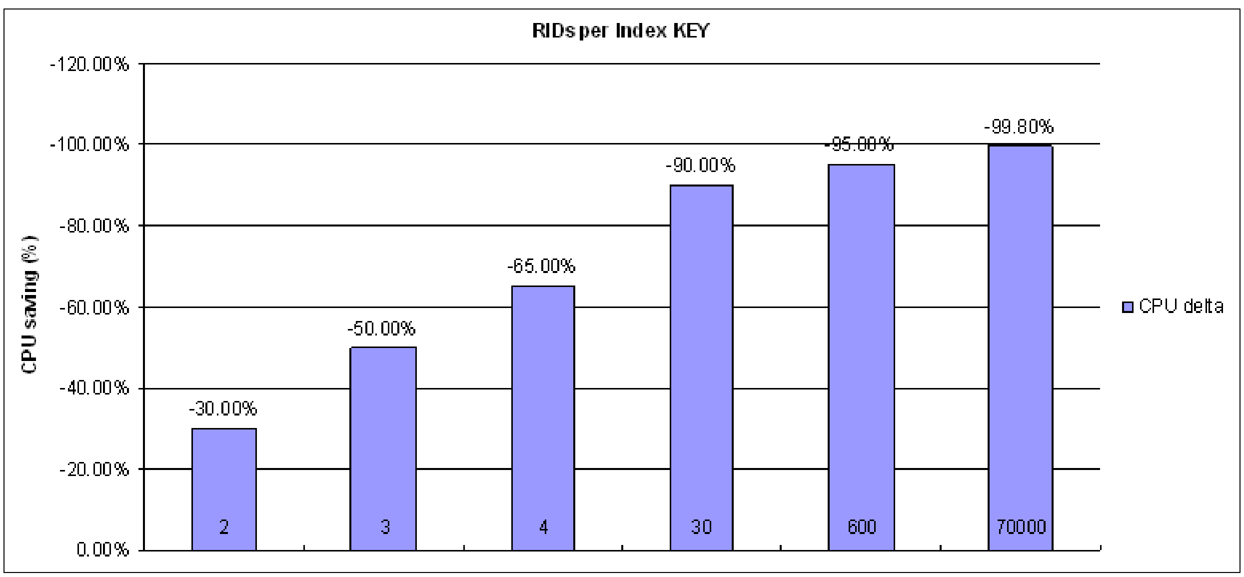

Figure 7-4 Reduction in DB2 11 CPU consumption with index duplicate skipping

DB2 11 can improve the performance up to 10,000 times for queries with an inner join that takes advantage of both the duplicate skipping and early out enhancements in DB2 11.

Table 7-1 on page 146 compares DB2 10 and DB2 11 performance. It shows DB2 10 and DB2 11 CPU consumption for multi-table GROUP BY and DISTINCT queries with a large number of duplicate index keys.

Table 7-1 Class 2 CPU in seconds for multi-table GROUP BY and DISTINCT queries

	DB2 10	DB2 11
RUNQ500	89.981	0.021
RUNQ510	600.825	0.021
RUNQ600	190.780	0.038
RUNQ610	223.535	0.037

Example 7-44 shows one sample of the queries that are used for the performance measurements to show these impacts.

Example 7-44 Inner join query

```
SELECT DISTINCT L_SUPPKEY FROM LINEITEM, PARTSUPP WHERE L_SUPPKEY=PS_SUPPKEY AND L_SUPPKEY
< 5000 ORDER BY L_SUPPKEY;
```

To enable this function, no user action is necessary.

There are two new DSN_DETCOST_TABLE Explain table columns to externalize information related to these new DB2 11 enhancements: columns IXSCAN_SKIP_DUPS and EARLY_OUT. They confirm whether DB2 11 duplicate skipping and early out are part of an access plan.

7.5 DPSI enhancements for queries

DB2 V8 introduced the ability to physically partition secondary indexes. Such indexes are called data partitioned secondary indexes (DPSIs) in contrast to the usual nonpartitioned secondary indexes (NPSIs). A NPSI normally consists of one single large data set, up to 64 GB in size. They can grow to more pieces, which are individual data sets, as needed. No matter how large the NPSI is, an NPSI has one index tree structure. In contrast, a DPSI, like the partitioning index, has one index tree structure for each partition, and each structure is contained in a different data set. DB2 can ensure key uniqueness for NPSI, but not for DPSI.

Why did DB2 V8 introduce DPSIs? Indexes may be created for several reasons: To enforce a uniqueness constraint, to achieve data clustering, but most likely, to provide access paths to data for queries or referential constraint enforcement. While the cost of maintaining any index must always be evaluated against its benefit, several unique factors come into play when deciding whether to add an NPSI to a table in a partitioned table space. This is because there are areas where NPSIs can cause performance and contention problems.

7.5.1 Availability considerations with NPSI and DPSI

Partitioned table spaces are recommended for storing large tables. Two of the reasons for the recommendation deal with availability issues:

- Potential to divide and conquer: The elapsed time to perform certain utilities, or the storage requirement to perform online REORG against a large table space, may be prohibitively high. Because utility jobs can be run at the partition level, operations on a table space can be broken along partition boundaries into jobs of more manageable size. The jobs may be run in parallel to accomplish the task in reduced elapsed time, or serially to limit resource that is consumed by the task at any one point.

- Positive recovery characteristics: If the data set backing a physical partition becomes damaged, the data outage is limited to that partition's data, and only that fraction of the data needs to be recovered. Furthermore, if partitioning is performed along lines meaningful to applications, any logical damage (by a bad behaving application, for example) can be isolated to certain partitions limiting the data outage and recovery scope.

These positive aspects of partitioning begin to deteriorate if there are NPSIs present:

- A partition level REORG needs to update the RIDs in each NPSI that correspond to that partition. Before DB2 11, these updates required synchronous reads of the NPSI to update the RIDs. DB2 11 provides improvements to avoid these synchronous reads, but the solution requires that the NPSI keys for the entire table space be sorted instead of sorting only the keys for the specific partitions being reorganized. Either way that DB2 would choose to update the NPSI, a DPSI provides a vast improvement to partition level REORG, DB2 only needs to sort the keys for the specific partitions being reorganized without doing any synchronous reads to the DPSI.
- LOAD PART jobs, running concurrently, contend on NPSIs because keys of all parts are interleaved. In addition, during a LOAD PART job, key processing against NPSIs follows insert logic, which is slower than append logic. DPSIs have no such problems with concurrency.
- Recovery from media failure is available for the entire index only. No piece-level rebuild or recovery is available. DPSIs can be copied and recovered at the partition level. Individual partitions can be rebuilt in parallel to achieve a fast rebuild of the entire index.

> **Important:** Since DB2 V8 DPSIs promote high availability by facilitating efficient **utility** processing on secondary indexes.

7.5.2 Partition-level operations

One common partitioning scheme is to partition by date. Typically, the oldest data resides in part 1 and the youngest data resides in part 2 to n. As time progresses and more data is collected, you need to add partitions to the table space to house the new periods of data. At some point enough history has been collected, and you now want to do one of two things:

- To discard the oldest partition and reuse that partition to collect the next period's data. This reflects a cyclic use of some set of partition numbers.
- To roll-off the oldest partition of data and roll-on a new partition in which to collect the next period's data. This delays the need to reuse a partition number until the name space is exhausted (that is until the number wraps). A variation on this is to make the data in “rolled off” partitions available to queries only on demand. The partition is hibernating, but can be awakened to participate in special queries.

Partition-level operations become more complex if there are NPSIs present.

For example, to erase the data of a partition, you normally LOAD that partition with an empty input file. The operation quickly “resets” the data partition, and the partitioned index if there is one, but also entails removing key entries from each NPSI that reference the partition being erased. For each row being removed from that partition, DB2 has to look up the key entry for that row in each NPSI and delete it. DPSIs have no such problem because DB2 can simply treat that data set as though it were a new data set.

7.5.3 Data sharing

In a data sharing environment, some customers find benefit from isolating the work on certain members to certain partitions.

Such affinity-routing eliminates intersystem read/write interest on physical partitions, thereby reducing data sharing overhead. Affinity routing does not alleviate contention on NPSIs since keys belong to different data partitions are spread throughout the NPSI. Also in this case, DPSIs can come to the rescue. Because page set P-locking occurs at the physical partition level, affinity routing is effective for DPSIs because the parts of the DPSI are aligned with data parts in the table. This is most beneficial during batch type applications with heavy insert/update/delete activity.

For a detailed description of DPSI, and the other index structures introduced in DB2 V8, see *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079.

7.5.4 DPSI query considerations

The physical nature of a DPSI can degrade the performance of some types of queries. Queries with predicates that solely reference columns of the secondary index are likely to experience performance degradation, due to the need to probe each partition of the index for values that satisfy the predicate. Queries with predicates against the secondary index that also restrict the query to a single partition (by also referencing columns of the partitioning index), on the other hand, benefit from the organization.

For example, if you are aware of a correlation between DPSI and partitioning index (PI) key values, you need to code the PI restriction explicitly when supplying a DPSI predicate to facilitate partition pruning.

Let us assume that the partitioning column of a table is DATE, and a DPSI exists on ORDERNO. If the company has a policy that the first four digits of ORDERNO are always a four-digit year, write queries on ORDERNO as queries on ORDERNO and DATE, with DATE restricted to reflect this policy. This allows the pruning of uninteresting partitions from the query.

The DB2 optimizer is aware of the nature of DPSIs and takes the strong points and weaknesses into account when determining the best access path.

> **Note:** DPSIs allow for query parallelism and are likely to be picked by the optimizer for queries with predicates on partitioning columns plus predicates on the secondary index columns.

This brings us to DB2 11, which has made considerable improvements to the performance of queries when using DPSI. These improvements are described in the following sections.

7.5.5 DB2 11 page range screening and indexing for DPSI

When local predicates exist on the partitioning columns, DB2 can limit the access to only the qualified partitions. In DB2 11, this support is extended to join predicates. In a partitioning scheme where the table is partitioned by columns used in queries as join predicates, DB2 11 can use those predicates to filter out unnecessary partitions and probe only the qualified parts. This enhancement is most effective when the index for the join is a DPSI and the partitioning columns are excluded from the index or not the leading index columns.

Before DB2 11, optimal join performance could be achieved for this partitioning scheme only in either of the following situations:

- The index was created as an NPI.
- The index was partitioned (a partitioning index (PI), not a DPSI), but the partitioning columns were the leading index columns.

Figure 7-5 shows a join between T1 and T2, where the inner table of the join (T2) uses a DPSI (on C1) and a join predicate exists on the non-indexed YEAR column. In this example, each probe to the inner table can use the join predicate on the partitioning column to ensure that only the necessary partition is accessed for each inner table access.

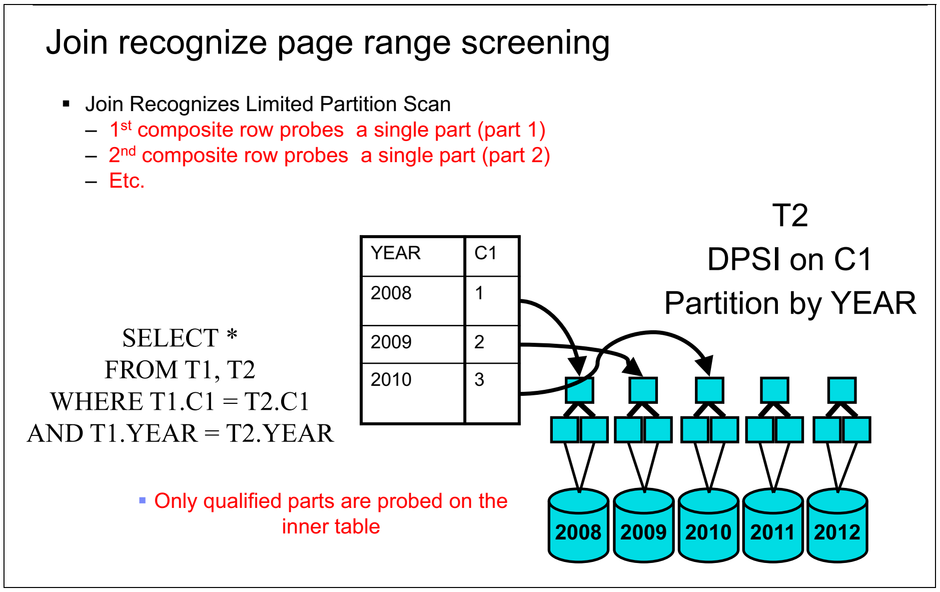

Figure 7-5 Page range screening from a join

Before DB2 11, each inner table access probed all partitions, resolving the YEAR join predicate after the data page was accessed. The benefit of the page range screening enhancement is that clients can convert to using a DPSI, which can ultimately enhance utility performance if fewer NPIs exist for a table.

To further expand the usage of DPSIs in DB2 11, additional enhancements are available for workloads that already use DPSIs or are considering converting more of their NPIs to DPSIs. However, although DB2 11 increases the use of DPSIs, still several scenarios exist where having one index b-tree structure (as with NPIs) has considerably better query performance than having one b-tree per partition.

7.5.6 DB2 11 page range partition ordering with DPSI

When a query contains ORDER BY, GROUP BY, or DISTINCT, and the query requires a subset of the table rows, an index is more efficient to provide that order rather than introducing a sort. For a partitioned table space, both a PI and an NPI can provide order that can match an ORDER BY, GROUP BY, or DISTINCT. However, a DPSI provides order only within a partition but not across the entire table space.

DB2 can use one of two methods to allow a DPSI to provide order without requiring a sort:

- Have parallelism provide order, where each child task processes one partition and the parent task merges the results to provide one ordered set.

- Use DPSI merge (also known as DPSI return in order), where DB2 processes each partition serially, but a merge process returns the result in order across all partitions.

The DPSI merge process is enhanced in DB2 11. First, index on expression can now use the DPSI merge; and second, DPSI merge has general performance enhancements such as improved index look-a-side and, thus, getpage avoidance.

DB2 11 adds concurrent DPSI I/O access to benefit very short running SQL, which use DPSI merge access. When multiple DPSI parts are probed one after another, DB2 11 supports concurrent I/O requests from qualifying DPSI parts in order to reduce elapsed times.

Important: DB2 11 provides index on expression support for DPSIs.

7.5.7 DB2 11 part level join parallelism with DPSI

The next enhancement to DPSIs involves using parallelism for improved join performance when the partitioned table space is the inner table of a join. To benefit from this enhancement, the partitioning scheme must be based on columns that are not included as join predicates in a query. Figure 7-6 illustrates the enhancement, which is referred to as a *part-level join*. In this example, the table is partitioned by YEAR (where each partition is numbered 2009 - 2013), although the query includes only join predicates on C1.

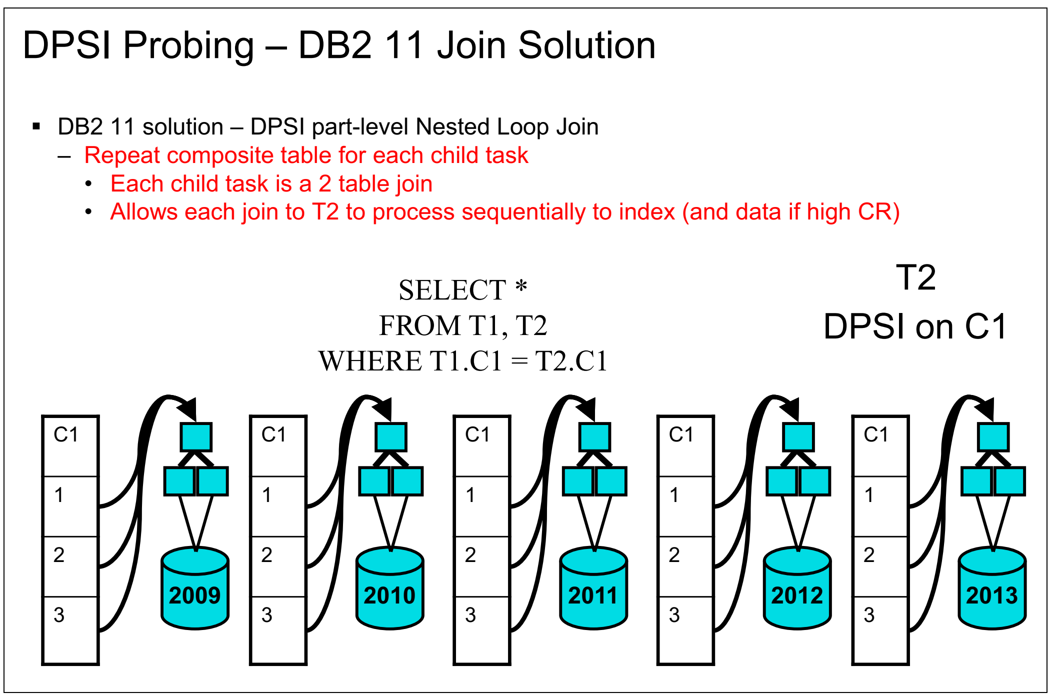

Figure 7-6 Part-level join

With the part-level join, each parallel child task processes only one partition, and the composite (outer) table is shared or replicated to each child task. This method allows each child task to act as though it is a two-table join (involving one partition) rather than a join to multiple partitions.

In DB2 10, an issue with DPSI join performance is that the join results in a large amount of random I/O because each partition is probed on the inner table of the join. In DB2 11, both page range screening from join predicates and part-level join should improve join performance for DPSIs. These enhancements might also potentially allow more workloads to convert NPIs to use DPSIs.

This function is effective in DB2 11 CM. To enable this function, a bind, rebind, or full prepare is necessary.

The PARAMDEG_DPSI subsystem parameter (MAXDEGREE FOR DPSI option on panel DSNTIP81) specifies the maximum degree of parallelism that you can specify for a parallel group in which a DPSI is used to drive parallelism. It sets the maximum parallel degree for DPSI parallelism independently from PARAMDEG (MAX DEGREE option in DSNTIP81).

Acceptable values: 0-254, DISABLE:

0 DB2 uses the value of the PARAMDEG subsystem parameter, instead of PARAMDEG_DPSI. This is the default value.

1 DB2 creates multiple child tasks but works on one task at a time when DPSI is used to drive parallelism.

2-254 DB2 creates multiple child tasks and works concurrently on the tasks that are specified. When PARAMDEG is set to 1, the rest of the query does not have any parallelism.

DISABLE DB2 does not use DPSI to drive parallelism. Parallelism might still occur if PARAMDEG is greater than 1.

7.5.8 DPSI measurements

Several measurements were done highlighting the areas of DPSI improvement in DB2 11 compared to DB2 10.

The DW10 benchmark (data warehousing sets of 22 queries examining large volumes of data with some degree of complexity) comparing DB2 10 to DB2 11 (converted to DPSI, NO compression) shows an average CPU reduction of 30% for DEGREE=ANY.

Figure 7-7 on page 152 shows the DW10 results:

- Up to a 50% reduction in CPU for certain SQL due to optimizer correctly recognizing the need for, and cost, of DPSI merge access.
- Reduced CPU and elapsed times for improvements in index lookaside and sequential detection when using DPSI access.
- 10% reduction in CPU and up to a 90% reduction in getpages for DPSI merge access.
- OLTP-type SQL shows a 50 - 90% reduction in elapsed times due to DB2 11's new parallel I/O support for multi-part DPSI merge access.
- Up to a 95% reduction in CPU and getpages for DPSI-correlated subquery access.

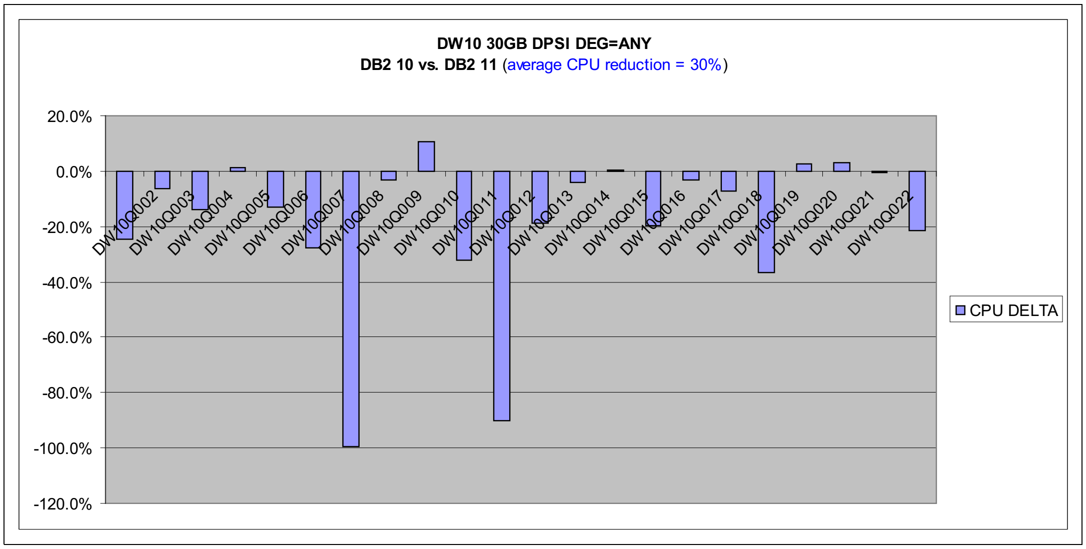

Figure 7-7 DW10 queries highlighting DB2 11 DPSI improvements

7.5.9 Conclusions

The decision to use an NPSI or DPSI must take into account both maintenance practices and the access patterns of the data. Replace an existing NPSI with a DPSI only if there are perceivable benefits such as easier data or index maintenance, or improved data or index availability, or improved performance.

DB2 11 enables DPSIs to perform queries much better than before, but beware of making wholesale changes from NPIs to DPSIs without careful consideration of the costs and benefits outlined in this section.

In general, if most of your data access is cases where local or join predicates limit the qualifying partitions to a small subset of the total parts, DPSIs might be a better choice than NPSIs.

7.6 LIKE predicate evaluation enhancement

DB2 10 generates machine code to evaluate most of the simple predicates to improve query performance significantly. The generation of machine code results in much more efficient custom code for each predicate. The machine code generation is done at BIND or PREPARE time, if possible, or dynamically at execution time.

DB2 11 improves the LIKE predicate processing on tables with (coded character set identifier) CCSID UNICODE, which was not supported in DB2 10.

The query in Example 7-45, with one LIKE predicate, was used to measure the predicate evaluation enhancement in DB2 11 on a table with CCSID UNICODE encoding with table space scan.

Example 7-45 Benchmark query with a single LIKE predicate

```
SELECT * from CUSTOMER
WHERE c_name LIKE 'Customer#0000000%'
```

The measurement showed up to 90% CPU time improvement without any access path changes in CM after REBIND.

7.7 Optimizer helping to enhance statistics

Collecting more complete and accurate statistics results in more accurately estimated selectivity, which results in improved access path choices. In general but specially with application packages, it is challenging to know what statistics to collect to obtain the best possible access path for your SQL statements. It requires that you have knowledge of each SQL statement.

DB2 11 provides an enhancement to externalize statistics recommendations for missing or conflicting statistics information during query optimization. The statistics recommendations can then be used to drive RUNSTATS such that DB2 has more accurate and complete statistics during query optimization and, as a result, can choose more efficient access paths.

In this section, we describe the following enhancements that are related to table space statistics:

- Statistics profiles
- Statistics cleanup
- Recognition and documentation of missing statistics

For details, see Chapter 13, “Performance” of *DB2 11 for z/OS Technical Overview*, SG24-8180.

7.7.1 Statistics profiles

Running proper RUNSTATS is crucial for the DB2 optimizer to ensure that accurate information is used for access path selection. Although running RUNSTATS is not a feature of DB2 that is devoid of DBA or user involvement, this is an area where an enhanced ease of use and automation features would be welcomed. DB2 11 takes steps to improve RUNSTATS in the areas of ease of use, and provides recognition of important missing statistics.

DB2 10 delivered statistics profiles, so that clients could combine the complexity of individualized statistics into a stored profile, and subsequent RUNSTATS executions could use that profile to ensure consistency of statistics collection. DB2 11 supports integrating those profiles into a LISTDEF control statement. Therefore, the USE PROFILE keyword can be added to the LISTDEF control statement. Tables with a profile collect their specialized statistics, and those tables without continue to collect the basic statistics.

7.7.2 Statistics cleanup

Simplified syntax in DB2 11 clears the statistics for a table and its associated index. The RESET ACCESSPATH keyword of RUNSTATS resets all statistics for the named objects and clears any specialized frequency, cardinalities, or histograms from the catalog. After the statistics are cleared, you can recollect the wanted statistics. This function is effective in DB2 11 CM.

7.7.3 Recognition and documentation of missing statistics

This RUNSTATS enhancement uses guidance that is provided by the DB2 optimizer about the statistics that were missing during a BIND or REBIND, dynamic PREPARE, or EXPLAIN.

While determining the access path, the DB2 optimizer externalizes whether the statistics that could be used by the optimizer are missing or conflicting.

This information is externalized to the DB2 catalog (SYSIBM.SYSSTATFEEDBACK catalog table) in DB2 11 New Function Mode or into a new explain table (DSN_STAT_FEEDBACK, if such a table exists), or both. A user can then use this information to determine which RUNSTATS to collect to potentially improve the optimizer's chance at choosing an optimal access path. Alternatively, tools such as IBM Optim™ Query Workload Tuner can interpret the information and provide the RUNSTATS input.

Externalizing the statistics recommendations as part of general query processing is a significant step forward compared to individually analyzing a query or having tools collect a workload for analysis.

7.8 RID pool overflow to work file

DB2 10 provided new techniques to better manage the RID pool and to overcome its limits by allowing an access plan to overflow to a work file and continue processing RIDs even when one of the RID thresholds is encountered at run time. However, SQL statements that used aggregate functions such as:

- COUNT()
- SUM()
- AVG()

were not eligible for work file overflow in DB2 10. DB2 11 extends this capability to these SQL statements.

At run time, if a RID threshold is exceeded, RIDs overflow to the work file and continue processing. Thus, RID access rarely needs to fall back to a table space scan (as with DB2 9). In some cases, you might see work file use increase as a result of RID list overflow to work file.

As expected, there is some overhead associated with overflowing RID requests to a work file. Measurements show less than a 5% increase in CPU time and less than a 10% increase in elapsed time. The increases in CPU and elapsed time can be larger if you have a constrained work file buffer pool, but is still less than the cost of falling back to a table space scan.

The RID pool overflow to a work file does not impact access path selection. The overflow to a work file is a run time decision.

Important: DB2 can still fall back to a table space scan if there is not enough work file storage for RID requests.

The maximum amount of work file storage that can be used for RID list processing is controlled by the DB2 system parameter MAXTEMPS_RID. If MAXTEMPS_RID is set to 0, RID pool overflow is unlimited. If it is set to a very small nonzero value, RID pool overflow is very limited. The default in DB2 11 is NOLIMIT and it is a good starting point.

Because RID list processing for hybrid joins does not overflow to a work file, in DB2 10 a large RID pool can provide some extra insurance against a few hybrid joins consuming your entire RID pool and causing all shorter running RID list access to overflow to a work file. DB2 11 limits the consumption of RID pool from hybrid join to 80%, therefore reducing such risk.

The RID pool work file overflow enhancement is available in DB2 11 CM. Rebind of applications is not required unless you have not rebound them in DB2 10 to enable rid overflow.

The following DB2 subsystem parameters related to this function exist:

- RID POOL SIZE field

  MAXRBLK subsystem parameter specifies the amount of storage in kilobytes that is needed for the RID pool. Default is 400000. You can estimate the storage that is required for the RID pool by using the following formula:

  ```
  Number of concurrent RID processing activities *
  average number of RIDs * 2 * 5 (bytes per RID)
  ```

- MAX TEMP RID field

  The MAXTEMPS_RID subsystem parameter determines the maximum amount of temporary storage in the work file database that a single RID list can use at a time. The default is NOLIMIT.

- MAX TEMP STG/AGENT field

  The MAXTEMPS subsystem parameter determines the maximum amount of temporary storage in the work file database that a single agent can use at any given time for any type of usage. The usage can be for sort work files, created global temporary tables, declared global temporary tables, scrollable cursors result tables. The default is 0 MB.

  If MAXTEMPS is set to a nonzero value, it overrides the setting of MAXTEMPS_RID.

7.9 Sort and in memory work file

The work files that are used in sort are logical work files, which reside in work file table spaces in your work file database.

The sort begins with the input phase, when ordered sets of rows are written to work files. At the end of the input phase, when all the rows have been sorted and inserted into the work files, the work files are merged together, if necessary, into a single work file that contains the sorted data.

A sort can complete in the buffer pool without I/Os operations if the amount of data being sorted is small. The sort row size is made up of the columns being sorted (the length of the sort key) and the columns that the user selects (the length of the sort data). A large buffer pool for sort activity can help avoid disk I/O operations.

When your application needs to sort data, DB2 gives priority to try to allocate each sort work file on a segmented (non-universal) table space with no secondary allocation.

This section describes these topics:

- Physical sort work file usage reduction
- Sort processing enhancement
- In-memory work file enhancement
- Reuse work file

7.9.1 Physical sort work file usage reduction

Sort processing can be a bottleneck for query performance. One of the costly operations in sort is using physical work files. DB2 11 enhances sort performance by reducing physical work file usage to save I/O and CPU time. There are three enhancements:

- Final sort merge process
- Single sort run
- Sparse indexes, range partitioning, and tag sorts

Final sort merge process

The DB2 11 *final sort merge* process avoids the allocation of physical work file for final sort merge process. When a final sort has to be materialized in a work file, it requires large work file allocation as the final merge needs to collapse multiple intermediate work files into one final big physical work file. This enhancement is only supported for top-query block-level sort and there is no parallelism, no cursor hold, and the work file is not being used for set function work file scan. Set functions are functions such as MIN, MAX, SUM, and COUNT.

Laboratory measurements showed that DB2 class 2 CPU time is reduced by 8% to 27% for GROUP BY sort and DB2 class 2 CPU time is reduced by 9% to 28% for top level ORDER BY sort with different sort record sizes.

Single sort run

The second enhancement, *single sort run*, avoids single output work file usage for final sort if data (`(sort key + data) * length`) does not exceed 1 MB and only one sort run is necessary. In this case, if sort has determined that all the records fit within the external nodes during the input phase and only one run is necessary through the tournament sort, instead of writing the sorted records to a final output work file, DB2 11 writes out the records directly to the user application. As a result, the creation of the final output work file and insertion of data and fetching from the work file is avoided. This can only be done when this is the top-level query sort and there is no parallelism, no cursor hold, and the work file is not being used for set function work file scan.

Laboratory measurements showed that DB2 class 2 CPU time was reduced by 7% - 16% for GROUP BY sort, and DB2 class 2 CPU time was reduced by 17% to 31% for top-level ORDER BY sort.

Sparse indexes, range partitioning, and tag sorts

The third enhancement avoids intermediate work file usage for sparse index, range partitioning, or tag sort. Before DB2 11, the design consisted on writing the output of sparse index, range partitioning, and tag sort into an intermediate work file during the input phase, and in rereading the work file content during the merge phase and then rewriting it to in-memory or work file. DB2 11 skips the phase to write into the intermediate work file to reduce the work file processing overhead. This only occurs during the input phase if sort knows that only one work file would have been created. This could happen on any sort process within the query.

Tag sort is used by sort when the sort record is over 4032 bytes. During the input phase, the data to be sorted is stored on a side work file. During the tournament sort, the sort record just contains the key+RID from the side work file, which contains much less data. After the input phase, a merge phase would be called to build the final output record to contain the actual data. The RID is used to help fetch this information when building this final output.

Laboratory measurements showed these performance improvements:

- DB2 11 reduced the DB2 class 2 CPU time by 2% to 27% for ORDER BY tag sort

- DB2 11 reduced the DB2 class 2 CPU time by 1.23% for sparse index with parallelism and 10% without parallelism
- DB2 11 reduced the DB2 class 2 CPU time by 1.21% for record range partitioning

7.9.2 Sort processing enhancement

DB2 11 improves general sort processing by creating generated code to build the sort external nodes for all the columns including data and key columns. This generated code is assembler code that gets generated within the sort module and is executed after each record is retrieved by run time. This enhancement is supported in DB2 11 CM mode and does not require any rebind of existing database applications.

DB2 query performance workload shows 1% to 7% CPU improvement without any query access path changes.

7.9.3 In-memory work file enhancement

The in-memory work file enhancement of DB2 11 provides performance improvements that help workloads with queries that use non-sort related work files. This enhancement facilitates wider usage of the in-memory work files above the 2 GB bar by keeping the record size < 32 KB and maximum number of rows < 255 in memory. Physical work file is used if the work file cannot fit in memory. DB2 11 also enhances the system availability by detecting system storage constraint before deciding to use the in-memory work.

The queries with non-sort related work file usage includes, not exclusively:

- Work file generated for sort merge join when inner table does not require SORT
- Work file generated for OUTER JOIN COMPOSITE TABLE result
- Work file created for Star Join snow-flake result
- Work file created for self referencing INSERT
- Work file created for materialized view or table expression if the last miniplan does not have a SORT and no select expression and no Data Manager evaluated SET function
- Work file created for EXISTS subquery if SORT cannot be built on the subquery

Laboratory measurements of queries with select from materialized table expressions with different number of rows and different number of row lengths materialized to the work files shows 2% to 58% CPU time reduction when the work file fits in memory with size of 32 KB. There is up to 2.62% CPU time reduction, and no performance regression, when the work files are spilled to use physical work files. The expected CPU time savings are bigger for small work file usage in a high concurrent environment due to the reduction of work file space map contentions.

7.9.4 Reuse work file

DB2 11 supports reuse of physical work file for repeated access for correlated subqueries. Previous to DB2 11, if the correlated subquery requires materialization work, files are created and destructed for every qualified row from the parent query block. Overhead of creating and destructing the work file becomes very high when the work file contains only 1 or a few rows.

This enhancement is limited to cases with correlated subqueries with SET (aggregation) functions, which require materialization and the size of the work fits in one 32 KB page.

Example 7-46 shows the DDL and a sample query used for benchmarking this change.

Example 7-46 Work file reuse scenario: DDL and Query

```
DDL:
CREATE TABLE CUSTOMER(BIRTHDATE DATE, CUSTNO INT);
CREATE TABLE CUSTOMER2(BIRTHDATE DATE, CUSTNO INT);
CREATE VIEW CUSTOMERS(BIRTHDATE, CUSTNO) AS
(SELECT * FROM CUSTOMER
UNION ALL
SELECT * FROM CUSTOMER2);

Query:
SELECT COUNT(*)
FROM CUSTOMERS C
WHERE BIRTHDATE =
 (SELECT MAX(C2.BIRTHDATE)
  FROM CUSTOMERS C2
  WHERE C2.CUSTNO = C.CUSTNO);
```

Qualified queries show 65% CPU reduction, and 81% CPU reduction when enabling query parallelism. Sideways reference queries show 64% CPU reduction. The correlated reference in a predicate in a correlated nested table expression is also referred to as a *sideways reference*. See Example 7-47 for an example of sideways query.

Example 7-47 Side way reference query

```
SELECT Orderline_Order_ID AS ORDER_ID,
       Orderline_District_ID AS dist_id,
       Orderline_Warehouse_ID AS wh_id,
       Orderline_Category AS categ,
       Orderline_Item_ID AS item_id,
       Orderline_Supply_Warehouse_ID AS s_ware_ID,
       Orderline_Quantity AS quanty,
       Orderline_Delivery_Date AS Delivery_date
  FROM Orderline T1, TABLE (
         SELECT DISTINCT(1)
           FROM Order  AS T3
           WHERE T3.Order_District_ID = T1.Orderline_District_ID
             AND T3.Order_District_ID = 'WA'
             AND T3.Order_ship_date = T1.Orderline_Delivery_Dat
```

7.10 EXPLAIN enhancements

You can call the ADMIN_EXPLAIN_MAINT stored procedure to create EXPLAIN tables, upgrade them to the format for the current DB2 11 release, or complete other maintenance tasks.

Most EXPLAIN tables have the additional column EXPANSION_REASON, which applies only to statements that reference archive tables or temporal tables.

DSN_PREDICAT_TABLE

The new predicate table, DSN_PREDICAT_TABLE, contains information about all of the predicates in a query. More values are provided for the ADDED_PRED column to indicate why DB2 added the predicate.

Whether the predicate is generated by DB2, and the reason why the predicate is added:

- Blank: DB2 did not add the predicate.
- 'B' For bubble up.
- 'C' For correlation.
- 'J' For join.
- 'K' For LIKE for expression-based index.
- 'L' For localization.
- 'P' For push down.
- 'R' For page range.
- 'S' For simplification.
- 'T' For transitive closure.

DSN_PREDICATE_SELECTIVITY

The new DSN_PREDICATE_SELECTIVITY table contains information about the selectivity of predicates that are used for access path selection. See 7.11, "Overriding predicate selectivities at the statement level" on page 160.

DSN_STAT_FEEDBACK

The new DSN_STAT_FEEDBACK table contains recommendations for capturing missing or conflicting statistics that are defined during EXPLAIN. Collecting these statistics by the RUNSTATS utility might improve the performance of the query.

DSN_USERQUERY_TABLE

The DSN_USERQUERY_TABLE table identifies statements whose access paths are influenced. The values identify the statements and the method that is used to influence access path selection. Values in the DSN_USERQUERY_TABLE are used to populate certain catalog tables when a `BIND QUERY` command is issued.

DSN_QUERY_TABLE

DSN_QUERY_TABLE contains query text for before and after query transformation. This table shows XML query rewrite in the same way that it does with other index types.

DSN_VIRTUAL_KEYTARGETS

The DSN_VIRTUAL_KEYTARGETS table contains information about expression-based indexes and XML indexes.

PLAN_TABLE

The PLAN_TABLE has new columns:

- SCAN_DIRECTION

  For index access, the direction of the index scan (F Forward, R Reverse)

- EXPANSION_REASON

And changed columns:

- MERGN (new values)

  Indicates whether the new table is consolidated before the join, or whether access that used a data partitioned secondary index (DPSI) involved a merge operation:

  – Y: Yes, the new table is consolidated before the join.
  – N: No, the new table is not consolidated before the join.
  – D: Access through a DPSI involved a merge operation.
  – U: Access through a DPSI that did not involve a merge operation.

- PRIMARY_ACCESSTYPE

  Indicates whether direct row access is attempted first:

  – D: DB2 tries to use direct row access with a rowid column. If DB2 cannot use direct row access with a rowid column at run time, it uses the access path that is described in the ACCESSTYPE column of PLAN_TABLE.

  – P: DB2 used data partitioned secondary index and a part-level operation to access the data.

  – T: The base table or result file is materialized into a work file, and the work file is accessed via sparse index access. If a base table is involved, ACCESSTYPE indicates how the base table is accessed.

DSN_PREDICAT_TABLE

The DSN_PREDICAT_TABLE has a new column:

- EXPANSION_REASON

And a changed column:

- ADDED_PRED (new values)

DSN_VIRTUAL_INDEXES

DSN_VIRTUAL_INDEXES has more columns to improve index modeling capabilities:

- KEY_TARGET_COUNT and IX_EXTENSION are similar to same column in SYSINDEXES. These columns describe the number and type of columns in the IX_EXTENSION table. For each row in DSN_VIRTUAL_INDEXES that represents the creation of a virtual index on expression corresponding rows need to be inserted in KEYTARGETS.
- UNIQUE_COUNT similar to same column in SYSINDEXES. The number of columns or key targets that make up the unique constraint of an index.
- IX_EXTENSION_TYPE is the type of extended index:
  – 'S' Index on a scalar expression
  – 'V' XML index.
  – Blank: A simple index.
- DATAREPEATFACTORF similar to same column defined in catalog: The anticipated number of data pages that are touched following an index key order. This statistic is only collected when the STATCLUS subsystem parameter is set to ENHANCED.
- SPARSE, similar to same column in SYSINDEXES Null suppressed indexes, allows an index to NOT have an index entry when all key columns that comprise the index contain the NULL value.

The Index Advisor function of the IBM Optim Query Workload Tuner tool has been enhanced to use the expanded virtual index capabilities.

7.11 Overriding predicate selectivities at the statement level

DB2 cannot estimate correctly filter factors for some types of predicates, such as:

- Host variables
- Parameter markers
- Expressions

- Table self-joins
- Subqueries

You can now override the default filter factors for certain predicates by creating selectivity overrides. Each predicate selectivity override describes the selectivity of a particular predicate in a particular SQL statement. When a statement contains more than one predicate, you can create separate selectivity overrides for each predicate in the statement.

Each set of overrides is a selectivity instance. Each selectivity instance has a weight value. The weight value represents the percentage of executions of the statement in which you expect the predicates to have the selectivities of that instance.

The selectivity instances for a single statement is called the *selectivity profile* of the statement. Like data statistics, selectivity overrides can also become stale over time as the data changes.

You can override the default selectivity estimates that DB2 uses to select access paths for SQL statements.

You need one of the following authorities:

- SQLADM
- SYSOPR
- SYSCTRL
- SYSADM

And need to create and access the EXPLAIN tables:

- DSN_USERQUERY_TABLE
- DSN_PREDICAT_TABLE
- DSN_PREDICATE_SELECTIVITY

Their DDL is in members DSNTESC and DSNTESH of the SDSNSAMP library.

The package that contains the statement has been created by a BIND PACKAGE statement.

7.11.1 Creating statement-level selectivity overrides

Populate DSN_PREDICAT_TABLE and DSN_PREDICATE_SELECTIVITY tables with values for the predicate selectivity. When you issue the **BIND QUERY** command, you can specify the input table schema to be used. The input can come from the following sources:

- Capturing EXPLAIN information

  DSN_PREDICAT_TABLE and DSN_PREDICATE_SELECTIVITY are populated with model rows that you can use to create selectivity overrides.

- Modifying the DSN_PREDICATE_SELECTIVITY table

  You can create a single or multiple selectivity instances for a predicate. You can create single instances by modifying the following values in the existing rows of the table.

If you want to create multiple selectivity instances, you must insert new rows for the additional instances.

The DSN_PREDICATE_SELECTIVITY table contains information about the selectivity of predicates that are used for access path selection. It is used as an input table for the **BIND QUERY** command when selectivity overrides are specified. It has the columns that are listed in Table 7-2 on page 162.

Table 7-2 DSN_PREDICATE_SELECTIVITY table QUERYNO

Column name	Type	Description
QUERYNO	INTEGER NOT NULL	A number that identifies the statement that is being explained. The origin of the value depends on the context of the row.
QBLOCKNO	SMALLINT NOT NULL	A number that identifies each query block within a query. The value of the numbers is not in any particular order, nor are they necessarily consecutive.
APPLNAME	VARCHAR(24) NOT NULL	The name of the application plan for the row. Applies only to embedded EXPLAIN statements that are executed from a plan or to statements that are explained when binding a plan. A blank indicates that the column is not applicable.
PROGNAME	VARCHAR(128) NOT NULL	The name of the program or package containing the statement being explained. Applies only to embedded EXPLAIN statements and to statements explained as the result of binding a plan or package. A blank indicates that the column is not applicable.
SECTNOI	INTEGER NOT NULL WITH DEFAULT	The section number of the statement. The value is taken from the same column in SYSPACKSTMT or SYSSTMT tables and can be used to join tables to reconstruct the access path for the statement. This column is applicable only for static statements. The default value of -1 indicates EXPLAIN information that was captured in Version 9 or earlier.
COLLID	VARCHAR(128) NOT NULL	The collection ID: DSNDYNAMICSQLCACHE The row originates from the dynamic statement cache DSNEXPLAINMODEYES The row originates from an application that specifies YES for the value of the CURRENT EXPLAIN MODE special register. DSNEXPLAINMODEEXPLAIN The row originates from an application that specifies EXPLAIN for the value of the CURRENT EXPLAIN MODE special register.
VERSION	VARCHAR(122) NOT NULL	The version identifier for the package. Applies only to an embedded EXPLAIN statement executed from a package or to a statement that is explained when binding a package. A blank indicates that the column is not applicable.
PREDNO	INTEGER NOT NULL	The predicate number, a number used to identify a specific predicate within a query.

Column name	Type	Description
INSTANCE	SMALLINT NOT NULL	The selectivity instance. Used to group related selectivities. Specify a value greater than 0 to identify the selectivity instance. Each instance can contain multiple predicates, but you cannot duplicate the same predicate within a single instance. By using the same value in several rows, you can group related predicates. You do not have to override every predicate for a statement. You can specify as many as 20 different instance values, if the sum of weight values for all instances equals 1.0. If the sum of weight values is less than 1.0, you can specify only 19 instance values. The INSTANCE values do not have to be consecutive.
SELECTIVITY	FLOAT NOT NULL	The selectivity estimate.
WEIGHT	FLOAT(4) NOT NULL	Specify a value 0 - 1 to indicate the weight of the selectivity instance. This WEIGHT value must match for every row that contains the same INSTANCE value. The sum of the WEIGHT values from multiple instances must not exceed 1. The recommendation is to weight each instance so that the sum of the weights is equal to 1. However, when the sum of the weights for multiple instances is less than 1, DB2 assumes the existence of another instance. The assumed instance uses the default selectivity assumption, and a weight of 1 minus the sum of the weights for the specified overrides.
ASSUMPTION	VARCHAR(128) NOT NULL	Indicates how the selectivity was estimated, or is used. One of the following values: 'NORMAL' Selectivity is estimated by using the normal selectivity assumptions. 'OVERRIDE' Selectivity is based on an override.
INSERT_TIME	TIMESTAMP NOT NULL GENERATED ALWAYS FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP	The time when the row was inserted or updated.
EXPLAIN_TIME	TIMESTAMP	The time when the EXPLAIN information was captured: All cached statements Non-cached static statements Non-cached dynamic statements
REMARKS	VARCHAR(762)	IBM use
EXPANSION_REASON	CHAR(2) NOT NULL WITH DEFAULT	This column applies to only statements that reference archive tables or temporal tables. For other statements, this column is blank.

When DB2 uses multiple selectivity instances to optimize a statement, the weighted averages of the selectivity instances are shown in the EXPLAIN cost estimates. The values that are reported in DSN_PREDICAT_TABLE also reflect the weighted-average filtering.

For details, see Chapter 38, “Influencing access path selection” of *DB2 11 for z/OS Managing Performance*, SC19-4060.

8

Application enablement

In this chapter, we describe the performance of DB2 11 for z/OS functions, not strictly confined to SQL, which provide infrastructure support for new applications or that simplify portability of existing applications to DB2 for z/OS from other database systems.

This chapter contains the following sections:

- Suppress NULL key
- Not logged declared global temporary tables
- Declared global temporary tables PREPARE and incremental bind avoidance
- XML optimizations
- Archive transparency
- NoSQL JSON support
- Global variables
- Grouping sets, ROLLUP, and CUBE support

8.1 Suppress NULL key

Some applications, especially general-purpose vendor packages, create indexes where every key contains a null value. A null key is defined to be a key where every column of the key contains a null value.

DB2 11 NFM can improve insert performance of NULL entries by the option of excluding NULL rows from indexes. The CREATE INDEX DDL statement is changed to state EXCLUDE NULL KEYS, and the RUNSTATS utility collect statistics only on non-NULL value.

All table statistics that are derived from an index are adjusted by the number of excluded NULL values. Therefore, the table statistics are the same whether they were derived from a table scan, an EXCLUDE NULL KEYS index, or a non-EXCLUDE NULL KEYS index (or INCLUDE NULL KEYS index). After converting existing indexes to EXCLUDE NULL indexes, monitor application performance. Insert performance should improve and query performance difference should be minimal.

When an index contains many null keys that are excluded from the index, performance is improved in the following ways:

- INSERT and DELETE performance is improved when a key is null. If the entire index entry is NULL, index insertion or deletion is skipped. UPDATE does not physically replace an index entry; rather the old entry is first deleted, and then the new entry key is inserted. UPDATE behaves just like INSERT and DELETE in that either can be skipped if the entire key value is NULL.
- Because NULL values are not indexed, access path selection avoids choosing an EXCLUDE NULL KEYS index whenever a NULL key could satisfy the predicate. The DISTINCT or NOT DISTINCT predicates and the NULL predicate do not filter NULL values, and are thus not sufficient by themselves to permit use of an EXCLUDE NULL KEYS index.
- During LOAD and REORG, index entries are extracted and placed in the sort work data set during the RELOAD phase. If a NULL index entry is detected, the index entry is never placed in the sort work data set. For SHRLEVEL CHANGE, the insert path is invoked and the logic to skip indexing a NULL index entry is the same as INSERT. SORTDATA NO cannot be specified when an EXCLUDE NULL KEYS index is the clustering index. Other utilities may also perform faster.
- Considerable DASD space is saved if most keys are null.

8.1.1 Performance measurements

To show the impacts, several tests were executed in this environment:

- DB2 11
- The table space type was partitioned by range with 100 million rows and 20 partitions
- The table was defined with two indexes. Both indexes were four-level indexes and the index keys were integers

  Only the second index contained null keys. In each instance, measurements were done where 5% of the keys were null, then 50%, and then 90%:

  - For 5% null keys, the index had 304,479 leaf pages
  - For 50% null keys, the index had 243,372 leaf pages
  - For 90% null keys, the index had 170,810 leaf pages
- System z z10 four-way processor

Utility performance

Figure 8-1 shows the CPU improvements for the DB2 utilities comparing EXCLUDE NULL KEYS to INCLUDE NULL KEYS. CPU improvements are shown in percentage.

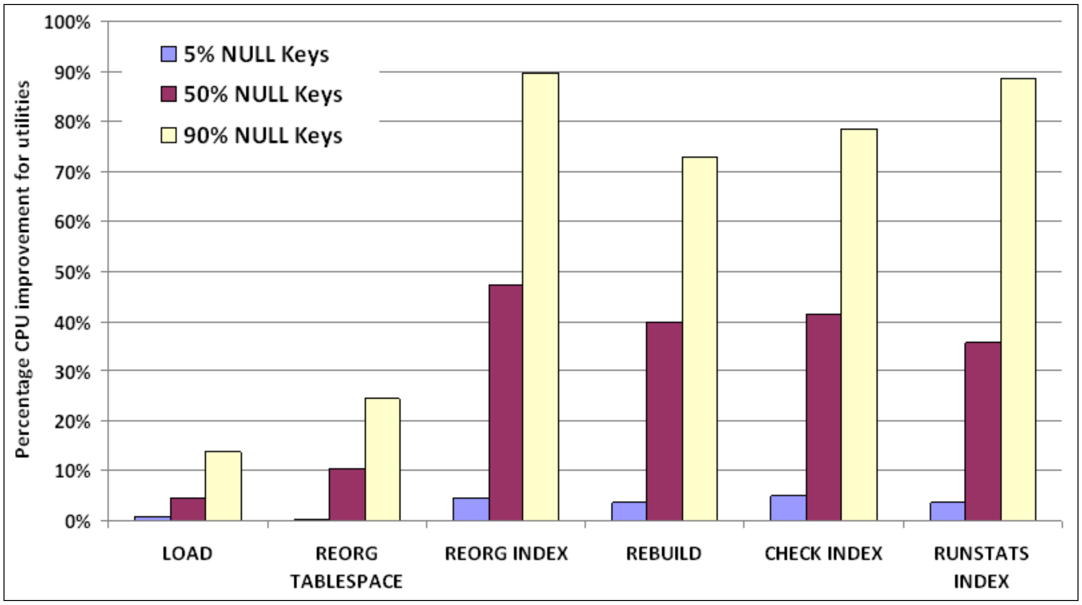

Figure 8-1 CPU improvements for the DB2 utilities: EXCLUDE NULL KEYS versus INCLUDE NULL KEYS

The utilities that target the index specifically, such as REORG INDEX, benefit the most from suppressing nulls. Utilities that operate against a table space, such as REORG TABLESPACE, do not benefit as much. The largest improvement measured was on REORG INDEX; when 90% of the keys were null, the CPU time was reduced by 90%. The biggest CPU improvement is observed in the SORT and BUILD phases for most utilities. The number of index getpages and updates was reduced up to 79% with 90% NULL keys.

Figure 8-2 shows both the CPU and elapsed time improvements for CREATE INDEX.

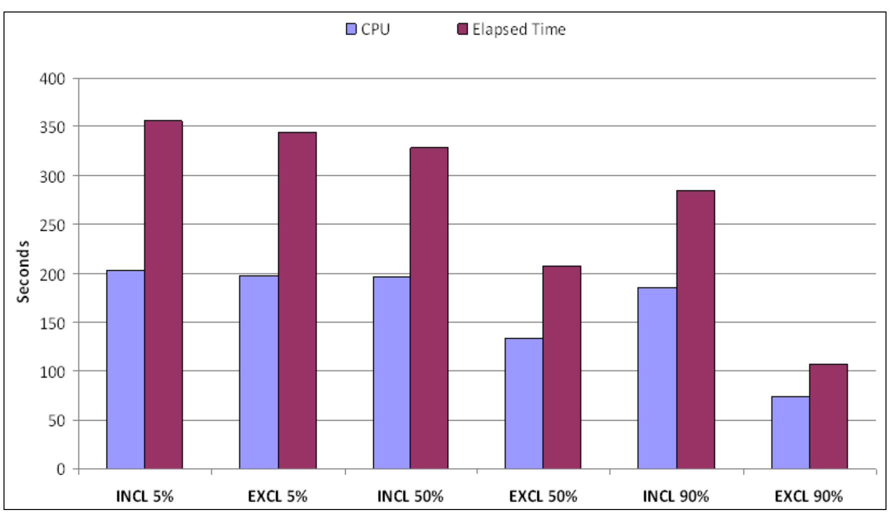

Figure 8-2 CPU and elapsed time improvements for CREATE INDEX

The number of index getpages and updates was reduced up to 79% with 90% NULL keys. The number of getpages and updates also reduce up to 50% in work file.

Figure 8-3 shows the resulting number of leaf pages.

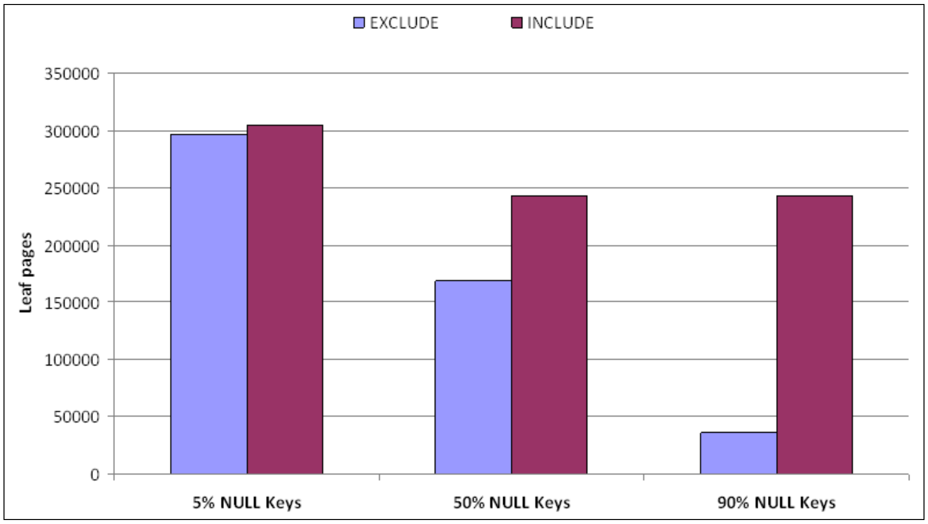

Figure 8-3 Space usage (Index) comparisons after LOAD

Insert performance in data sharing environment

Another set of measurements was designed to demonstrate the effect of suppressing NULLs on insert performance. Highly concurrent inserts were measured in this environment:

- DB2 11
- System z eight-way z196 processor
- Two-way data sharing environment
- Page level locking
- Multi-row inserts of 100 rows per SQL statement
- Commits were done after every 300 rows
- There were three PBG tables each with two indexes. Only the second index key contained nulls
- For the first table, the second index key contained an integer column and for the other two tables the second index key contained a non-padded VARCHAR(1000) column

Suppress null indexes were evaluated with 5%, 50%, and 90% of the keys being NULLs. Figure 8-4 on page 169 shows the throughput improvements: with 90% null keys, throughput increased by 45%.

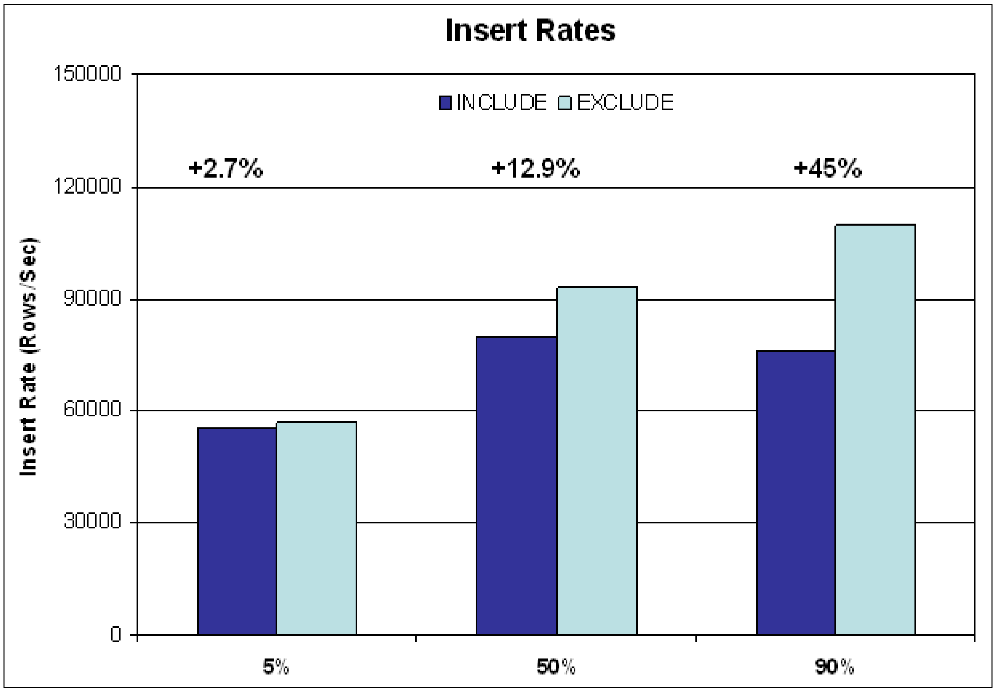

Figure 8-4 Insert rate comparison high insert workload for suppress null keys

Figure 8-5 shows the CPU reductions; with 90% null keys, the CPU time was reduced by 38%.

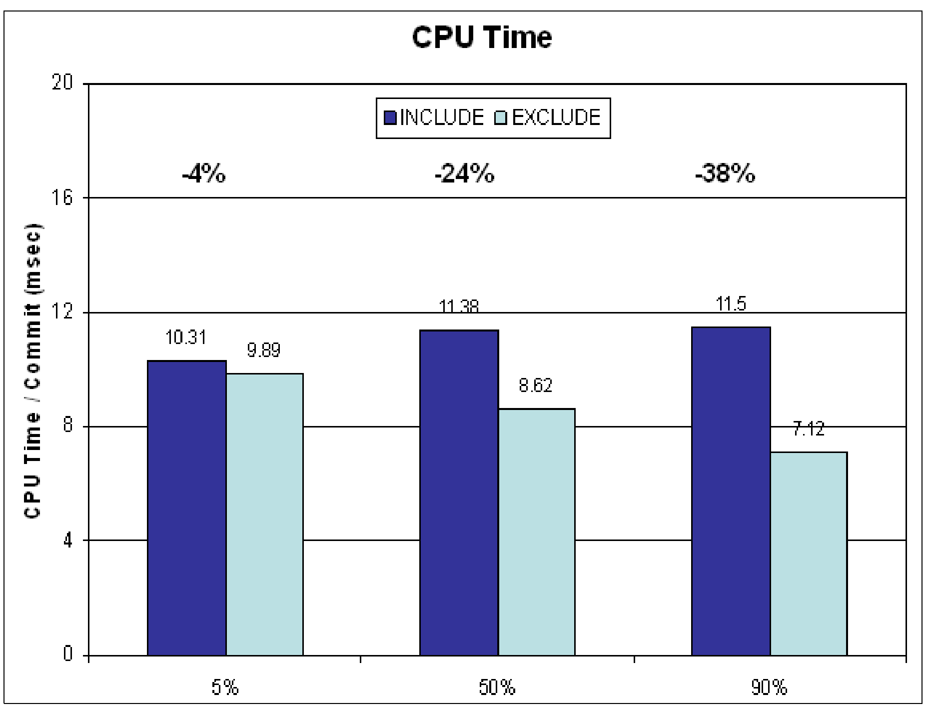

Figure 8-5 CPU time comparison high insert workload for suppress null keys

Figure 8-6 shows the number of leaf pages and number of active pages; with 90% null keys, both were reduced by 51%. In this chart, NLEAF is the number of leaf pages in the index, and NACTIVE is the number of active pages in the index space.

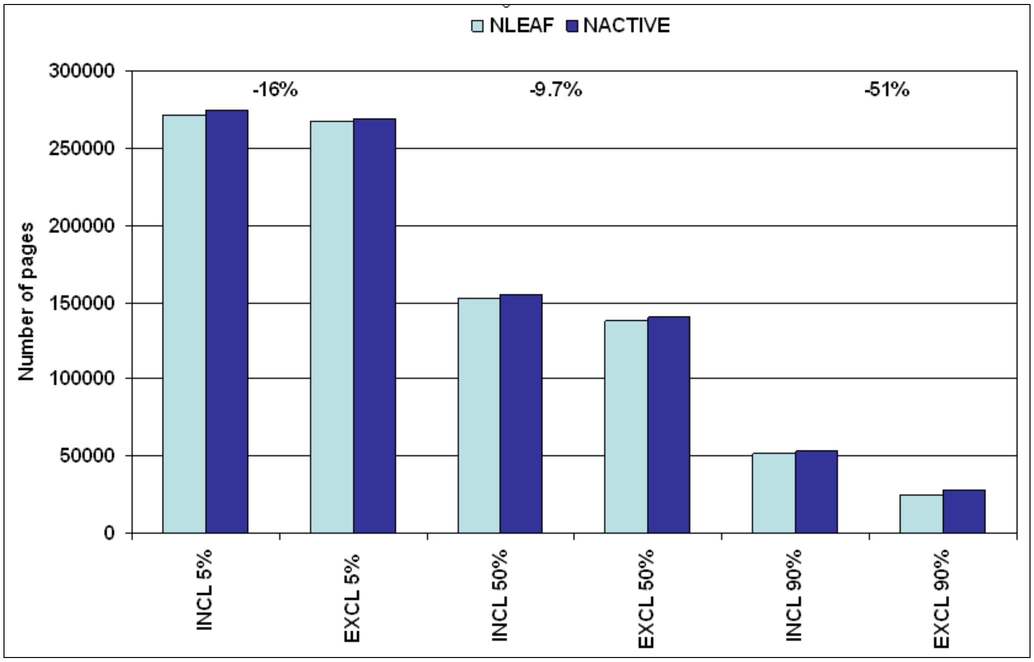

Figure 8-6 Space usage (index) comparisons: High insert workload for suppress null keys

Insert performance in non-data sharing environment

In this scenario, 5 million rows were inserted into a single PBG table in a non-data sharing environment. The second index key was a non-padded VARCHAR(1000) column. The workload consisted of single row inserts, 100 rows per commit. The objects were defined with page level locking, and no think time was included in the test.

The test showed that with 90% of the keys being null, suppressing the null keys achieved a 14.7% increase in throughput and a 13.2% reduction in CPU time.

Figure 8-7 on page 171 shows the insert rate results.

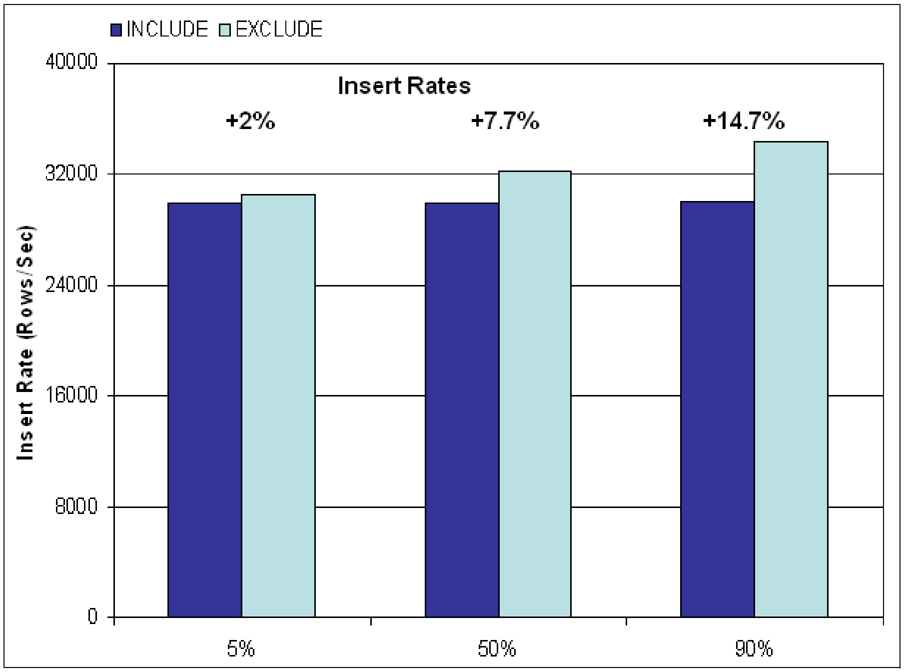

Figure 8-7 Insert rate comparison batch insert workload for suppress null keys

Figure 8-8 shows the CPU time results.

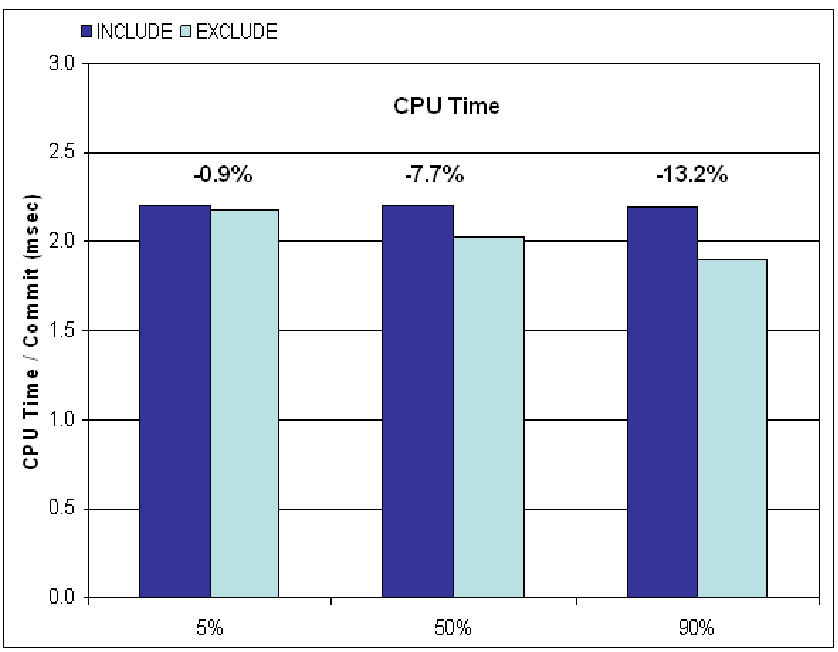

Figure 8-8 CPU time comparison batch insert workload for suppress null keys

Figure 8-9 shows that with 90% of the index keys being null, the number of leaf pages were reduced by 86%. In this chart, NLEAF is the number of leaf pages in the index, and NACTIVE is the number of active pages in the index space.

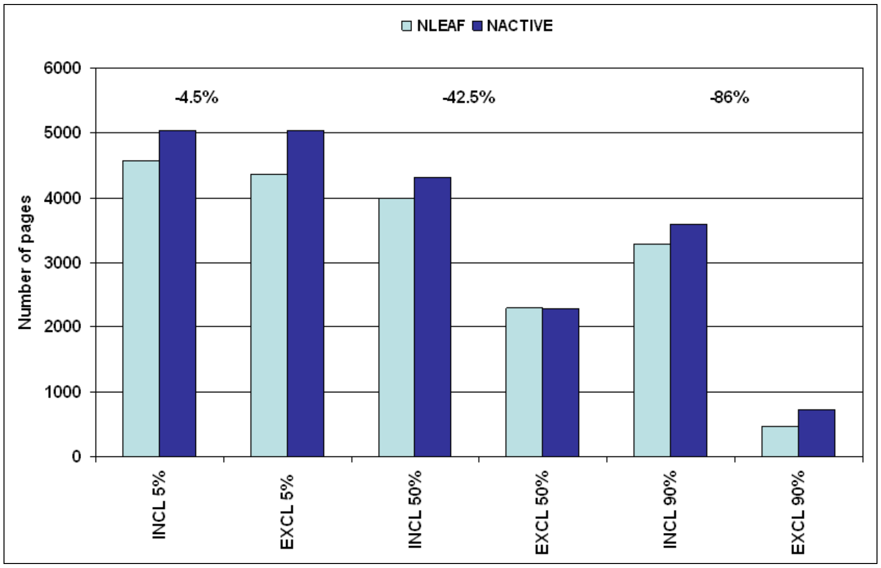

Figure 8-9 Space usage (index) comparisons: Batch insert workload for suppress null keys

Query performance

Finally, query performance was evaluated in the case where 90% of the index keys were null. A 9% CPU reduction was observed for a matching index scan and a 26% CPU reduction was observed for a non-matching index scan.

8.2 Not logged declared global temporary tables

Declared global temporary tables (DGTTs) are used by applications to store intermediate SQL result data while an application is still running. The implementation of DGTTs is improved in DB2 11 by allowing the user to avoid logging during insert, update, and delete activity to DGTTs.

This improves the performance and usability of DGTTs. With the ability to choose to not log DGTT activity, it may be beneficial to use DGTTs instead of created global temporary tables (CGTTs). While CGTTs support not logging and provide better performance as the schema is known before the application run, CGTTs do not support certain key features such as indexes.

This function also provides DB2 family compatibility because DB2 for Linux, UNIX, and Windows already supports not-logged DGTTs.This function is effective in DB2 11 NFM.

8.2.1 Enabling NOT LOGGED

With DB2 11, DGTT has the following logging options:

- LOGGED

  In this case, DB2 logs all changes, and during ROLLBACK or ROLLBACK TO SAVEPOINT, the changes to the DGTT are undone. This option is the default.

- NOT LOGGED ON ROLLBACK DELETE ROWS:

  This option specifies that you do not want logging to occur for this table, and during ROLLBACK or ROLLBACK TO SAVEPOINT, all rows in the DGTT are deleted if any change was made since the last COMMIT.
- NOT LOGGED ON ROLLBACK PRESERVE ROWS

  This option specifies that you do not want logging to occur for this table, and during ROLLBACK or ROLLBACK TO SAVEPOINT, the rows in the DGTT are preserved as they are.

We also talk about rollbacks when we think of error situations when there is a negative SQLCODE or message returned to the application. This kind of rollback is not affected by the new behavior. In the case of an error situation during an SQL statement, where an SQLCODE or message is issued, if an update was made to a DGTT and LOGGED is specified, the changes to the DGTT are undone.

In the case of an error situation during an SQL statement, where an SQLCODE or message is issued, if an update was made to a DGTT and NOT LOGGED is specified, all rows in that DGTT are deleted regardless of the DELETE/PRESERVE ROWS qualification.

Important: A decision on the ROLLBACK behavior for DGTTs only affects situations in which your application explicitly issues a ROLLBACK statement.

8.2.2 Performance measurements

This section describes the performance measurements conducted to measure the CPU impacts of this change, and when accessing a few partitions out of a large number of partitions in DB2 11.

The environments consisted of the following components:

- zEnterprise 196 with eight dedicated processors.
- z/OS 1.13 and DB2 for z/OS V11.
- DB2 DGTT table with up to 10 partitions and 53 columns including VARCHAR, BIGINT, DECIMAL AND TIMESTAMP.
- Insert 5 million rows from full select.

Example 8-1 lists the DDL and DML used for the measurements.

Example 8-1 DGTT DDL and DML

```
CREATE TABLE "ORDER_HISTORY"  (
    "ORDER_ID"              BIGINT      NOT NULL ,
    "EVENT_ID"              INTEGER     NOT NULL ,
    "MACHINE_ID"            VARCHAR(32) NOT NULL ,
    "SERVER_ID"             VARCHAR(32) NOT NULL ,
    "HON"                   INTEGER ,
    "ORIG_HON"              INTEGER ,
    "CORR_HON"              INTEGER ,
    "ACCOUNT"               VARCHAR(12) NOT NULL ,
    "SUBSCRIBER_ID"         CHAR(8)     NOT NULL ,
    "TRADER_ID"             VARCHAR(8)  NOT NULL ,
    "PRODUCER_ID"           VARCHAR(20) NOT NULL ,
    "PRODUCT_GROUP"         VARCHAR(5)  NOT NULL ,
    "PRODUCT_ID"            VARCHAR(20) NOT NULL ,
    "FIRM_ID"               VARCHAR(12) ,
```

```
    "OPERATOR_ID"           VARCHAR(32) ,
    "LOCATION_ID"           VARCHAR(32) ,
    "TON"                   CHAR(8) ,
    "CLIENT_ORDER_ID"       VARCHAR(32) ,
    "CORR_CLIENT_ORDER_ID"  VARCHAR(32) ,
    "ORIG_CLIENT_ORDER_ID"  VARCHAR(32) ,
    "ORDER_TYPE"            VARCHAR(20)   NOT NULL ,
    "ORDER_SIDE"            VARCHAR(5)    NOT NULL ,
    "TRANSACTION_TIME"      VARCHAR(30) ,
    "SENDING_TIME"          VARCHAR(30) ,
    "CTI_CODE"              VARCHAR(5) ,
    "CTI_ORIGIN"            VARCHAR(20)   NOT NULL ,
    "OMNIBUS_ACCOUNT"       CHAR(10) ,
    "TIME_IN_FORCE"         VARCHAR(20) ,
    "ORDER_STATUS"          VARCHAR(20) ,
    "REQUEST_STATUS"        VARCHAR(20) ,
    "ORDER_PRICE"           DECIMAL(31,8) NOT NULL ,
    "ORDER_QTY"             INTEGER       NOT NULL ,
    "REMAINING_QTY"         INTEGER       WITH DEFAULT 0 ,
    "TOTAL_FILL_QTY"        INTEGER       WITH DEFAULT 0 ,
    "FEE_TYPE"              VARCHAR(5) ,
    "GIVEUP_INDICATOR"      VARCHAR(20) ,
    "GIVEUP_FIRM_ID"        VARCHAR(3) ,
    "MARKET_STATE"          VARCHAR(20) ,
    "CREATION_TIME"         TIMESTAMP     NOT NULL ,
    "MSG_ID"                VARCHAR(1000) NOT NULL ,
    "MSG_SEQ"               INTEGER       NOT NULL ,
    "DISPLAY_TIME"          TIMESTAMP     NOT NULL ,
    "POST_TRADE_ALLOCATION" VARCHAR(5) ,
    "ENG_RECEIVED_TIME"     TIMESTAMP     NOT NULL ,
    "DB_SEND_TIME"          TIMESTAMP     NOT NULL ,
    "REJECT_REASON"         VARCHAR(100) ,
    "SESSION_DATE"          VARCHAR(8) ,
    "EXEC_ID"               DECIMAL(16,0) ,
    "QUOTE_TYPE"            VARCHAR(20)   WITH DEFAULT 'ORDER' ,
    "QUOTE_ID"              VARCHAR(10) ,
    "QUOTE_SET_ID"          VARCHAR(3) ,
    "QUOTE_ENTRY_ID"        VARCHAR(10) ,
  "REQ_ISINRFQ_ID"    VARCHAR(23) )
  PARTITION BY RANGE (EVENT_ID ASC)
    (PARTITION 1 ENDING AT (   1600000) ,
     PARTITION 2 ENDING AT (   3200000) ,
     PARTITION 3 ENDING AT (   4800000) ,
     PARTITION 4 ENDING AT (   6400000) ,
     PARTITION 5 ENDING AT (   8000000) ,
     PARTITION 6 ENDING AT (   9600000) ,
     PARTITION 7 ENDING AT (  11200000) ,
     PARTITION 8 ENDING AT (  12800000) ,
     PARTITION 9 ENDING AT (  14400000) ,
     PARTITION 10 ENDING AT(  16000000) )
   IN FALCDB01.TSFALC03;

DECLARE GLOBAL TEMPORARY TABLE SESSION.TEMPTAB1 LIKE
USRT001.ORDER_HISTORY
NOT LOGGED ON ROLLBACK DELETE ROWS;
```

```
--NOT LOGGED ON ROLLBACK PRESERVE ROWS;

INSERT INTO SESSION.TEMPTAB1
    SELECT * FROM USRT001.ORDER_HISTORY;
COMMIT;
```

Figure 8-10 shows the results when comparing both logged and not logged DGTT performance with 5 million rows inserted in DB2 11. It shows that there is more than 20% DB2 class 2 elapsed and CPU time savings with NOT LOGGED DGTT.

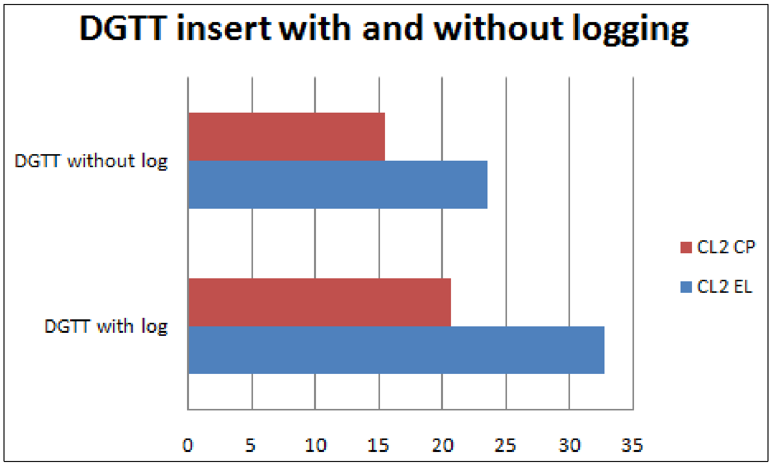

Figure 8-10 Insert performance: Logged versus not logged DGTT

Figure 8-11 shows that there is more than 20% DB2 class 2 elapsed and CPU time savings when applying NOT LOGGED on DGTT with 5 million rows deleted in DB2 11.

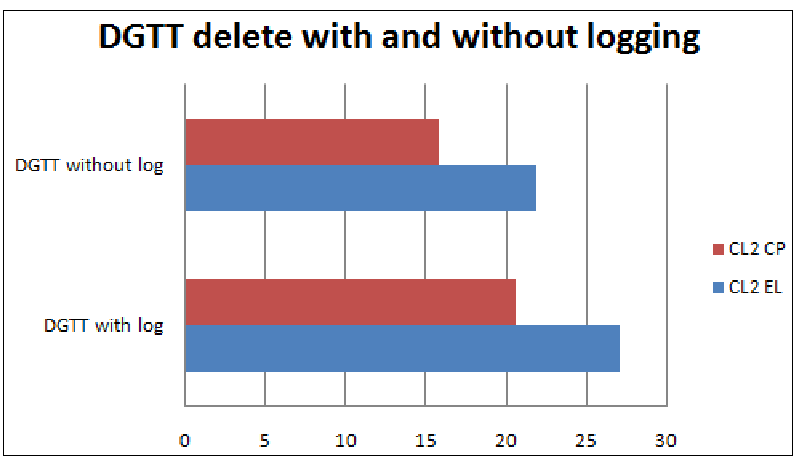

Figure 8-11 Delete performance: Logged versus not logged DGTT

Figure 8-12 shows that there is more than 20% DB2 class 2 elapsed and CPU time savings when applying NOT LOGGED on DGTT with 5 million rows updated in DB2 11.

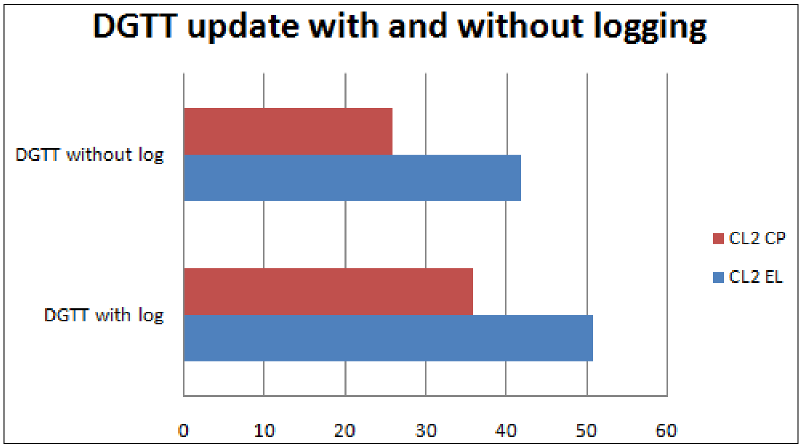

Figure 8-12 Update performance: Logged versus not logged DGTT

Figure 8-13 shows that there is more than 20% DB2 class 2 CPU time and 60% class 2 elapsed time reduction when applying NOT LOGGED ON ROLLBACK DELETE ROWS and NOT LOGGED ON ROLLBACK PRESERVE ROWS on DGTT respectively with 5 million rows insert change rolled back in DB2 11.

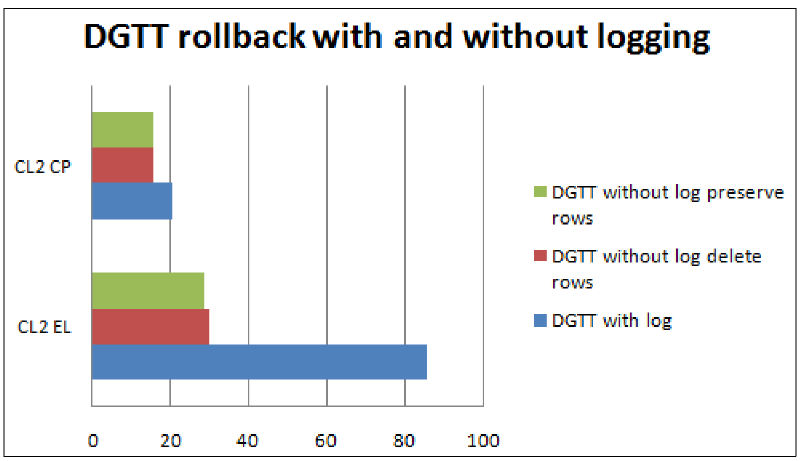

Figure 8-13 Rollback performance: Logged versus not logged with delete and preserve rows DGTT

8.3 Declared global temporary tables PREPARE and incremental bind avoidance

DB2 11 improves the performance of using DGTTs with certain SQL by removing the need for incremental bind (for static SQL) or re-PREPARE (for dynamic SQL) after COMMIT. The SQL statement is kept in an executable ready state past COMMIT so that subsequent statement execution does not need incremental bind or another PREPARE. This occurs when the application process is bound with RELEASE(DEALLOCATE) and for dynamic SQL, the unnecessary repeated PREPARE is removed.

Example 8-2 shows a sample OMPE accounting report highlighting the section with the "incremental bind" counter. This enhancement shows reduction for that counter for applications that benefit.

Example 8-2 Incremental bind avoidance counter

```
HIGHLIGHTS
--------------------------
#OCCURRENCES    :       35
#ALLIEDS        :        0
#ALLIEDS DISTRIB:        0
#DBATS          :       35
#DBATS DISTRIB. :        0
#NO PROGRAM DATA:        0
#NORMAL TERMINAT:       35
#DDFRRSAF ROLLUP:        0
#ABNORMAL TERMIN:        0
#CP/X PARALLEL. :        0
#UTIL PARALLEL. :        0
#IO PARALLELISM :        0
#PCA RUP COUNT  :        0
#RUP AUTONOM. PR:        0
#AUTONOMOUS PR  :        0
#INCREMENT. BIND:        0 QXINCRB
#COMMITS        :    42323
#ROLLBACKS      :      373
#SVPT REQUESTS  :        0
#SVPT RELEASE   :        0
#SVPT ROLLBACK  :        0
MAX SQL CASC LVL:        4
UPDATE/COMMIT   :     1.55
SYNCH I/O AVG.  : 0.000755
```

The performance improvement is most noticeable for very short running SQL using a DGTT and frequent COMMITs. To demonstrate the performance, a series of tests are run and the results shown in the charts following. The tests each comprise a DGTT, with and without and index, dynamic and static SQL, INSERT with and without a base table reference, with a COMMIT following each INSERT. Each individual test has 101 INSERT per COMMIT sequences. Each individual test is repeated 100 times and the average times are presented.

Example 8-3 shows the OMPE side by side comparison of excerpts from accounting report results showing how full prepares are reduced for dynamic SQL referencing a DGTT with intermittent COMMITs and for incremental binds for static SQL referencing a DGTT with intermittent COMMITs.

Example 8-3 Accounting report results showing full prepares

```
Sample OMPE excerpt from test of DGTT for dynamic and static SQL

            DB2 10                                         DB2 11
DYNAMIC SQL STMT       AVERAGE     TOTAL      DYNAMIC SQL STMT       AVERAGE     TOTAL
--------------------  --------  --------      --------------------  --------  --------
REOPTIMIZATION            0.00         0      REOPTIMIZATION            0.00         0
NOT FOUND IN CACHE      100.00     10000      NOT FOUND IN CACHE        0.00         0
FOUND IN CACHE            0.00         0      FOUND IN CACHE            0.00         0
IMPLICIT PREPARES       100.00     10000      IMPLICIT PREPARES         0.00         0
```

```
PREPARES AVOIDED          0.00        0     PREPARES AVOIDED          0.00        0
CACHE_LIMIT_EXCEEDED      0.00        0     CACHE_LIMIT_EXCEEDED      0.00        0
PREP_STMT_PURGED          0.00        0     PREP_STMT_PURGED          0.00        0
CSWL - STMTS PARSED       0.00        0     CSWL - STMTS PARSED       0.00        0
CSWL - LITS REPLACED      0.00        0     CSWL - LITS REPLACED      0.00        0
CSWL - MATCHES FOUND      0.00        0     CSWL - MATCHES FOUND      0.00        0
CSWL - DUPLS CREATED      0.00        0     CSWL - DUPLS CREATED      0.00        0

HIGHLIGHTS                                  HIGHLIGHTS
--------------------------                  --------------------------
#OCCURRENCES    :      100                  #OCCURRENCES    :      100
#ALLIEDS        :      100                  #ALLIEDS        :      100
#ALLIEDS DISTRIB:        0                  #ALLIEDS DISTRIB:        0
#DBATS          :        0                  #DBATS          :        0
#DBATS DISTRIB. :        0                  #DBATS DISTRIB. :        0
#NO PROGRAM DATA:        0                  #NO PROGRAM DATA:        0
#NORMAL TERMINAT:      100                  #NORMAL TERMINAT:      100
#DDFRRSAF ROLLUP:        0                  #DDFRRSAF ROLLUP:        0
#ABNORMAL TERMIN:        0                  #ABNORMAL TERMIN:        0
#CP/X PARALLEL. :        0                  #CP/X PARALLEL. :        0
#UTIL PARALLEL. :        0                  #UTIL PARALLEL. :        0
#IO PARALLELISM :        0                  #IO PARALLELISM :        0
#PCA RUP COUNT  :      N/A                  #PCA RUP COUNT  :        0
#RUP AUTONOM. PR:      N/A                  #RUP AUTONOM. PR:        0
#AUTONOMOUS PR  :      N/A                  #AUTONOMOUS PR  :        0
#INCREMENT. BIND:    10200                  #INCREMENT. BIND:      200
#COMMITS        :    10100                  #COMMITS        :    10100
#ROLLBACKS      :        0                  #ROLLBACKS      :        0
#SVPT REQUESTS  :        0                  #SVPT REQUESTS  :        0
#SVPT RELEASE   :        0                  #SVPT RELEASE   :        0
#SVPT ROLLBACK  :        0                  #SVPT ROLLBACK  :        0
MAX SQL CASC LVL:        0                  MAX SQL CASC LVL:        0
UPDATE/COMMIT   :     1.00                  UPDATE/COMMIT   :     1.00
SYNCH I/O AVG.  :      N/C                  SYNCH I/O AVG.  : 0.000948
```

For the dynamic case, the first INSERT requires a full prepare but the next 100 executions of the INSERT in DB2 10 require an “implicit prepare”, which is basically a full prepare. In DB2 11 the implicit prepares on each INSERT execution are eliminated. For the static SQL case in DB2 10, there is 1 incremental bind for the DGTT and 101 incremental binds for the INSERT SQL executions. In DB2 11 only the DGTT and the first INSERT execution require incremental binds.

Figure 8-14 on page 179 provides a comparison of class 2 CPU of eight different test configurations of DGTT insert/COMMIT activity. For a short running SQL statement, reducing implicit prepares and incremental binds can improve performance by a factor of 10.

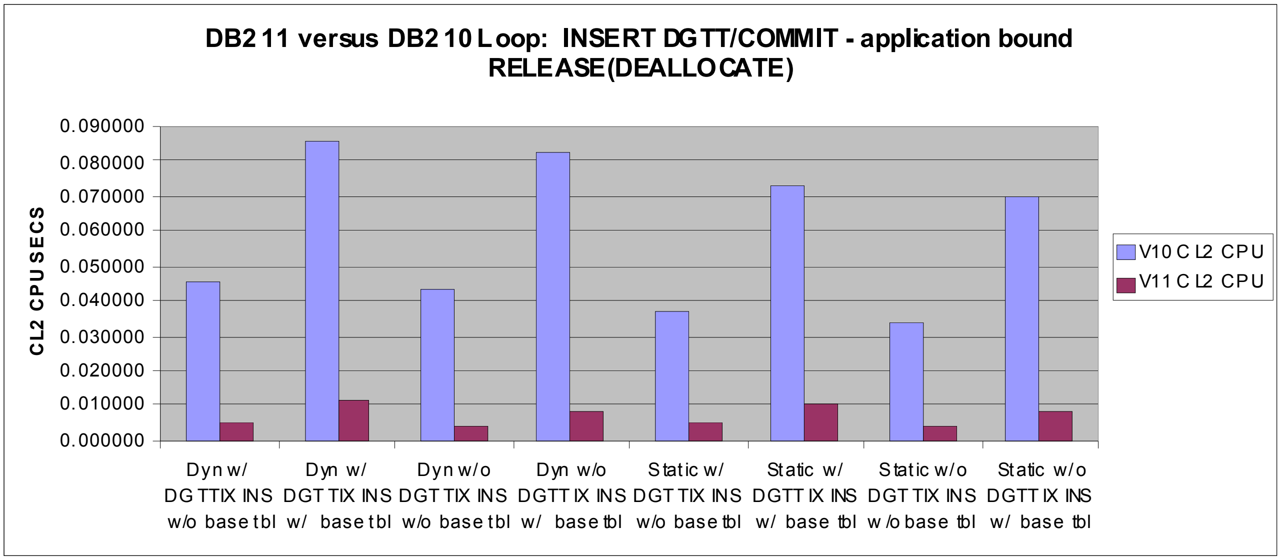

Figure 8-14 Test configurations of DGTT insert/COMMIT activity

8.4 XML optimizations

DB2 11 improves the performance of XML processing. Several performance enhancements are also retrofitted to DB2 10. See APAR II14426 for a list of current XML maintenance and a link for searching DB2 APARs containing XML:

http://www-01.ibm.com/support/search.wss?rs=64&r=100&cs=utf-8&rankfile=0&cc=&coll=0&spc=&stc=&apar=include&q1=XML&sort=desc&tc=SSEPEK&ibm-search.x=18&ibm-search.y=11&dc=DB550+D100&dtm

In the following sections, we describe the performance enhancements to XML in DB2 11, along with some measurements for the following:

- XQuery enhancement
- XMLTABLE: Optimize index key for VARCHAR predicates
- Extensible dynamic binary XML
- Avoid LOAD revalidation
- Partial validation

8.4.1 XQuery enhancement

The capability to store and access XML documents in DB2 for z/OS using an XML data type was first introduced in DB2 9. The initial implementation of XML in DB2 made use of the XPath language, which is a language for navigating XML documents and addressing parts of XML documents. XPath is a subset of XQuery, which is a richer language for accessing XML documents.

Although you can code SQL statements to access XML documents in DB2 9, the XPath language makes it difficult to write meaningful queries against XML data. In addition, the XPath language support in DB2 for z/OS is a small subset of the XQuery language that is supported on DB2 for Linux, UNIX, and Windows, which makes porting applications from DB2 for Linux, UNIX, and Windows to DB2 for z/OS difficult. As a result, you likely faced lost productivity due to the following issues:

- Rewriting XQuery queries to use syntax that DB2 supported

- Using a mixture of XPath and SQL/XML, which can be difficult to express
- Some query semantics not supported by XPath on DB2
- Challenges porting applications between different DB2 family members

The XQuery language support provided in DB2 11, and retrofitted to DB2 10 with APARs PM47617 and PM47618, allows application programmers to express such semantics and avoid unnecessary query rewrites. You can now spend less time switching back and forth between XQuery and SQL/XML and can express queries purely using XQuery instead. Differences between XQuery language support in DB2 for z/OS and DB2 for Linux, UNIX, and Windows still exist, but now in DB2 for z/OS, you can use commonly used XQuery language features, such as **`for`**, **`let`**, constructors, and **`if-then-else`**.

Refer to *DB2 11 Technical Overview,* SG24-8180 for details about DB2 11 support of XML.

DB2 11 introduced these performance enhancements for FLWOR expressions and XQuery support:

- Allows XQuery to push the predicates from an XQuery expression to an XPath expression.
- Translation of simple FLWOR expressions to XPath.

We ran a set of single-threaded XQueries with FLWOR expressions and different predicates.

Consider the following FLWOR expression:

```
for $i in /order/items/item
where $i/price > 100
return $i/desc
```

DB2 pushes the WHERE predicate into XPath as follows:

```
for $i in /order/items/item[price > 100]
return $i/desc
```

Another performance improvement is that DB2 can translate a simple FLWOR expression to XPath. For example, the following FLWOR expression:

```
for $i in /order/shipTo return $i
```

Can be rewritten into an XPath expression as follows:

```
/order/shipTo
```

Figure 8-15 on page 181 shows XQuery FLWOR single threaded query measurements. They show that these enhancements can provide up to 60% CPU time improvements for qualified queries.

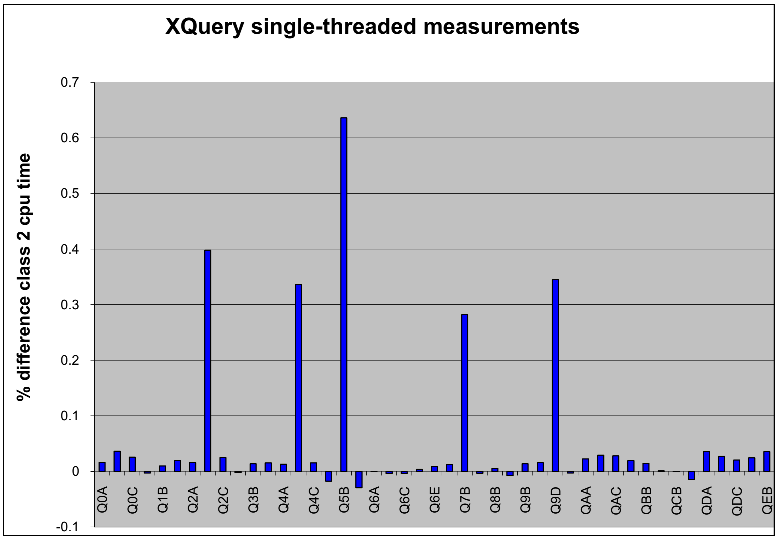

Figure 8-15   XQuery FLWOR: Single threaded measurements

8.4.2 XMLTABLE: Optimize index key for VARCHAR predicates

Before DB2 11, an XMLTABLE function that returns the results of an XQuery expression as a table, and with a column in that XMLTABLE function defined as VARCHAR, there was no way to specify an upper limit to the values that could qualify for the query.

DB2 11 adds an extra upper bound predicate to reduce the index key range after REBIND.

For our performance tests, we ran 100,000 single-threaded SELECT SQL statements with XMLTABLE and pushdown '=' predicates on both TIMESTAMP and VARCHAR indexes, as shown in Example 8-4.

Example 8-4   SELECT SQL statements with XMLTABLE and pushdown

```
SELECT ID FROM USRT012.N281DTB,XMLTABLE
('declare default element namespace
http://www.fixprotocol.org/FIXML-4-4
 /FIXML/Order'
 PASSING X1
 COLUMNS ID VARCHAR(10) PATH '@ID') X
WHERE X.ID = CAST (? AS VARCHAR(10))
```

Figure 8-16 shows the improvements in CPU time when using TIMESTAMP and VARCHAR predicates.

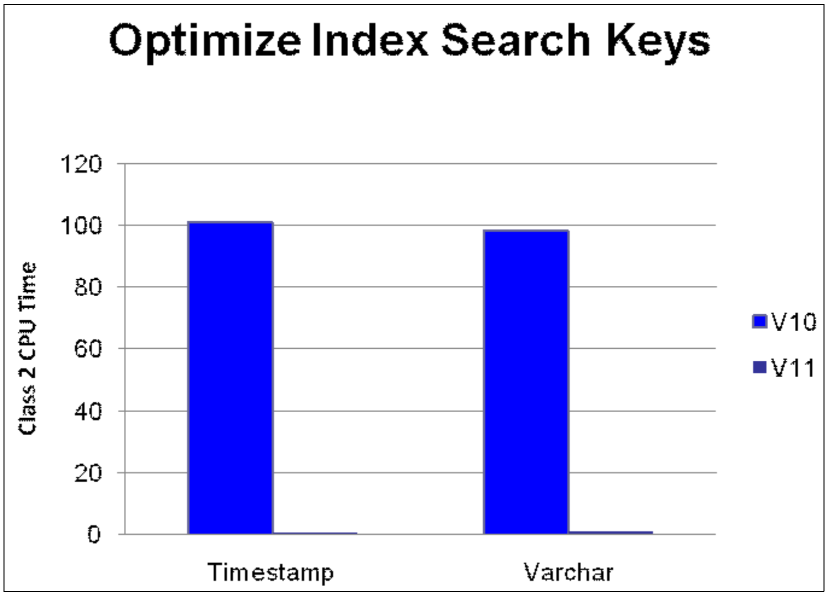

Figure 8-16 CPU time with TIMESTAMP and VARCHAR predicates

Figure 8-17 shows the improvements in total number of getpages when using TIMESTAMP and VARCHAR predicates.

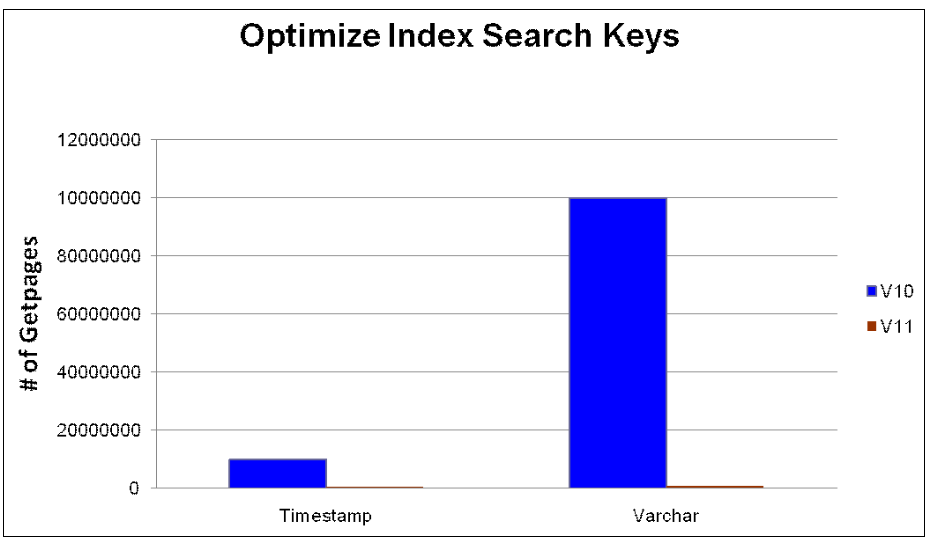

Figure 8-17 Total number of getpages with TIMESTAMP and VARCHAR predicates

8.4.3 Extensible dynamic binary XML

XML schema validation is the process of determining whether the structure, content, and data types of an XML document are valid according to an XML schema.

Before DB2 11, when a user sent over binary XML for schema validation, DB2 needed to convert the binary XML into textual XML before executing schema validation. In DB2 11, and DB2 10 with APAR PM53282, DB2 can directly validate binary XML at both Insert and Load times. This enhancement allows DB2 to avoid converting the XML document to text and do a true extensible dynamic binary (XDBX) validation. To take advantage of the XDBX parsing z/OS R13 PTF (PTF# UA63422) is needed.

XDBX schema validation performance

To test the performance of the XDBX schema validation enhancements in DB2 11, we ran tests to measure the XDBX schema validation cost with various sized documents against textual XML validation compared with the old method of binary XML validation (binary to textual conversion).

The following test scenarios were used:

- Load testing: load 500,000 rows of 4 - 10 KB sized XML documents
- Insert testing: insert various sized XML documents from industry standard workloads

There is no new syntax for this functionality. The savings are a result of the XDBX parser in z/OS. To insert using XDBX, your data just has to be in binary format and you insert with the same statement as normal XML. Here are some examples:

Unload data in binary form:

```
UNLOAD TABLESPACE DB2.T2;

 FROM TABLE T2;
    (ID POSITION(*) INT,
     XML1 POSITION(*)  XML BINARYXML )
```

Create a table with an XMLMODIFY column (schema defined for validation)

```
CREATE TABLE T1
   (ID INT, XML1 XML(XMLSCHEMA ID SYSXSR.custacc));
```

Load data back in binary XML:

```
LOAD DATA
 INTO TABLE
 "USRT014".
 "CUSTACCB"
 NUMRECS               500000
 ( ID POSITION(*)INT, XML1 POSITION(*) XML  BINARYXML )
```

Using the same DDL, the INSERT statement would be:

```
INSERT INTO T1(ID, XML1) VALUES(?, ?);
```

Figure 8-18 on page 184 shows a 22.7% elapsed time improvement when comparing the enhanced XDBX validation method against the binary to text method of validation. The measurement also shows 6.82% elapsed time savings when comparing XDBX to textual XML validation.

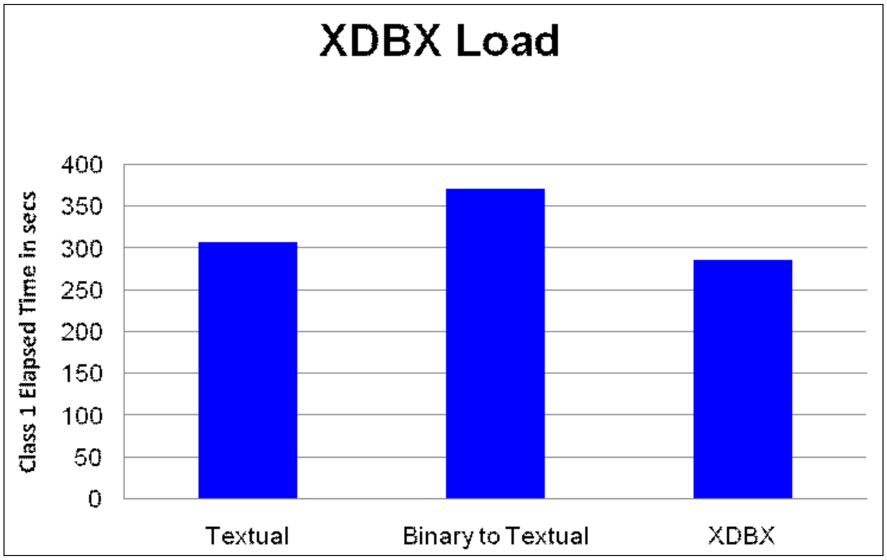

Figure 8-18 Load elapsed time

Figure 8-19 shows a 41.85% CPU time improvement when comparing the enhanced XDBX validation method against DB2 10 binary to text method of validation. The measurement also shows 18.91% CPU time savings when comparing XDBX to textual XML validation.

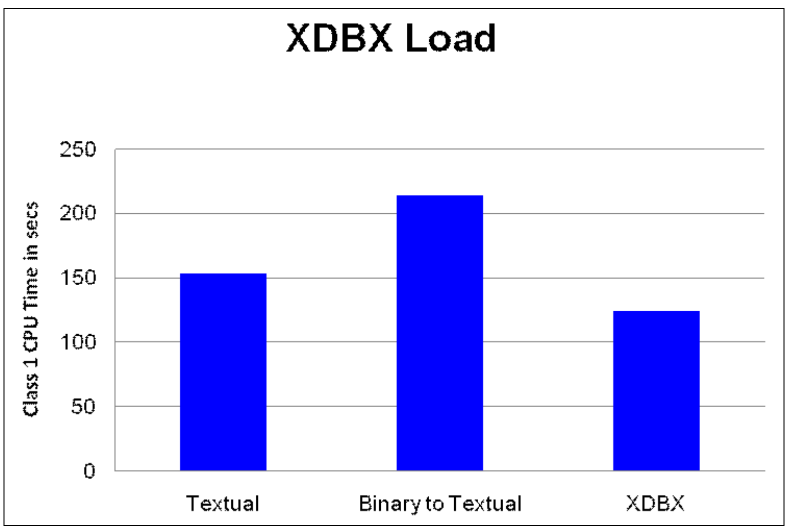

Figure 8-19 Load CPU Time

The measurements for our Insert tests are shown in Figure 8-20 on page 185 and Figure 8-21 on page 185. The bars labeled “XML” are for textual XML files, while the bars labeled “Binary” are for binary XML files converted to text before validation, and “XDBX” represents the enhanced XDBX validation. All sets of measurements were performed in DB2 11.

Figure 8-20 on page 185 shows up to a 30% class 2 elapsed time improvement when comparing XDBX to the binary to textual method and 28% savings when compared to textual XML validation.

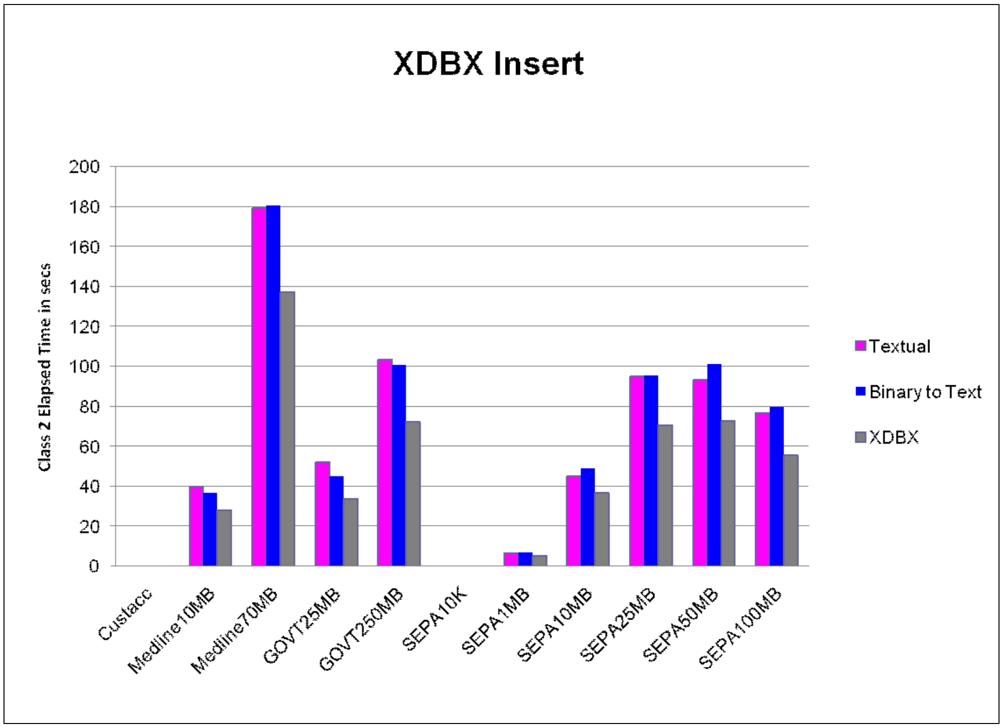

Figure 8-20 Insert class 2 elapsed time

Figure 8-21 shows that INSERT processing for binary XML documents using the enhanced XBDX validation consumed 30 - 40% less class 2 CPU when compared against the previous method of binary to text XML validation and 15 - 18% less CPU when compared to textual XML validation.

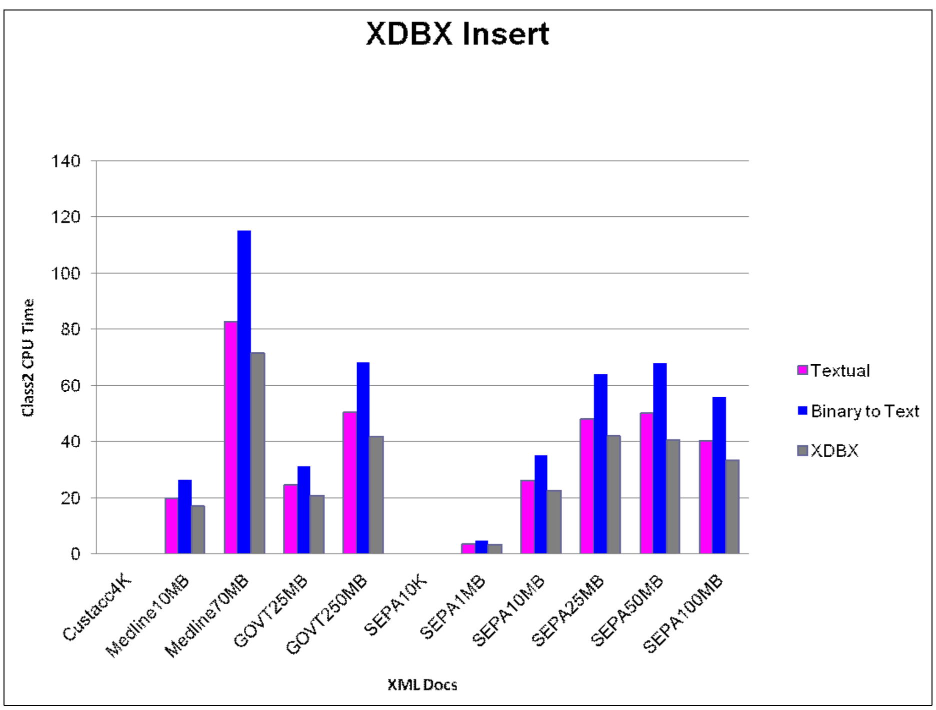

Figure 8-21 Insert class 2 CPU time

Sending Extensible Dynamic Binary XML DB2 Client/Server Binary XML Format data from a Java application on a client reduces the CPU time needed on the DB2 server. However, parsing the XML data into binary form is handled by the IBM Data Server Driver for JDBC and SQLJ running on the client.

Class 1 elapsed time on the client might increase when compared to sending textual XML. Use binary XML to reduce CPU time on the DB2 server if the increase of elapsed time does not impact your environment.

The XDBX enhancement is available in new function mode and in DB2 10 with the appropriate APAR applied. XDBX validation is also eligible for System z Integrated Information Processor (zIIP) off load. To take advantage of the XDBX parsing z/OS 1.13 APAR OA36712 (PTF UA63422) is needed.

The improvements are expected to be larger for larger documents.

8.4.4 Avoid LOAD revalidation

Before DB2 11, when you unloaded binary XML data and loaded it back, DB2 would revalidate the XML data against the registered schema. Starting with DB2 11, it is possible to skip the revalidation of the once loaded binary data.

Performance measurements

To measure the benefits of skipping validation of XML at load time, we ran the following tests using a mix of different-sized XML documents based on industry standard workloads:

- SEPA (10 KB, 1 MB, 10 MB, 25 MB, 50 MB, and 100 MB document sizes)
- Custacc (4 - 10 KB document sizes)
- Medline (10 MB document size)
- Govt (25 MB, 250 MB, and 766 MB document sizes)

Figure 8-22 shows that XML skip validation during LOAD provides 30 - 43% elapsed time improvements, depending on the document size.

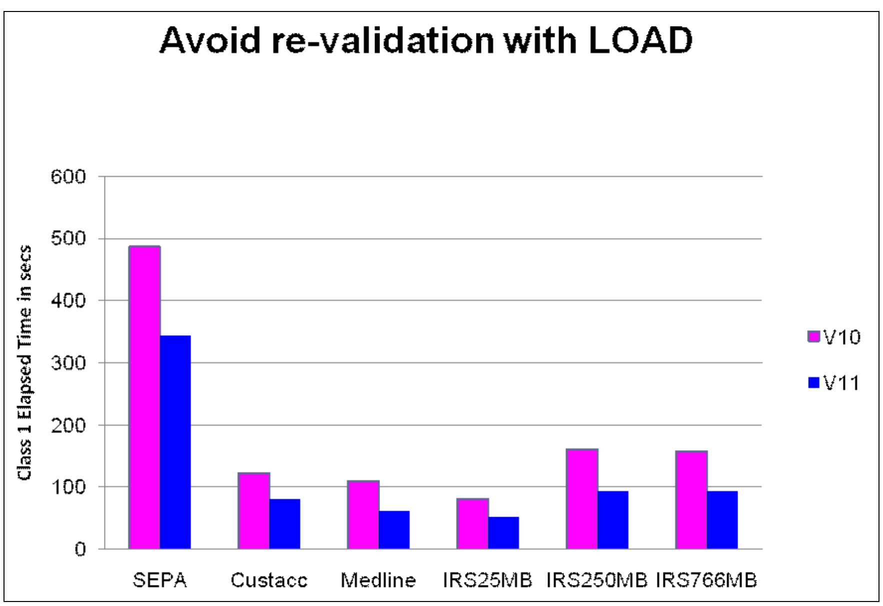

Figure 8-22 Skip validation during LOAD: class 1 elapsed time

Figure 8-23 shows that XML skip validation during LOAD provides 61 - 76% CPU time reduction in class 1 CPU time, depending on the workload and document size.

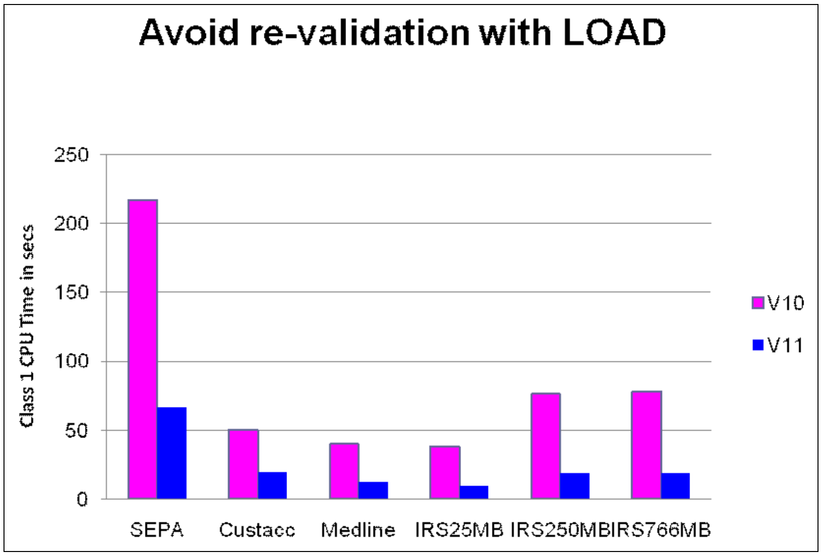

Figure 8-23 Skip validation during LOAD: class 1 CPU time

Overall, the XML schema skip validation enhancements can provide a significant reduction in DB2 class 1 CPU and elapsed time by skipping the revalidation process. The XML LOAD skip validation enhancement is available in DB2 11 new function mode.

8.4.5 Partial validation

Before DB2 11, when subdocument updates (XMLMODIFY) are applied to an XML type modifier column (XML column associated with 1 or more XML schemes), DB2 needs to revalidate the partially modified XML document before inserting it back to the XML column. Before V11, DB2 revalidated the whole document.

In DB2 11, and in DB2 10 with APAR PM69178 (PTF UK90369), DB2 only validates the partially modified XML document before inserting it back to the XML column.

XMLMODIFY partial validation performance

To test the performance of partial validation in DB2 11, we ran tests to measure the XML schema validation costs for various sized XML documents when updating, inserting, and deleting only portions of the document.

The measurements applied to the following types of workloads and document sizes, which are based on industry standard workloads:

- Custacc (4 - 10 KB document sizes)
- Medline (10 MB document size)

Figure 8-24 shows that, depending on the size of the XML document and the updated portion, we observe an elapsed time reduction of 58 - 93% for Update, 49 - 88% for Delete, and 52 - 93% for Insert.

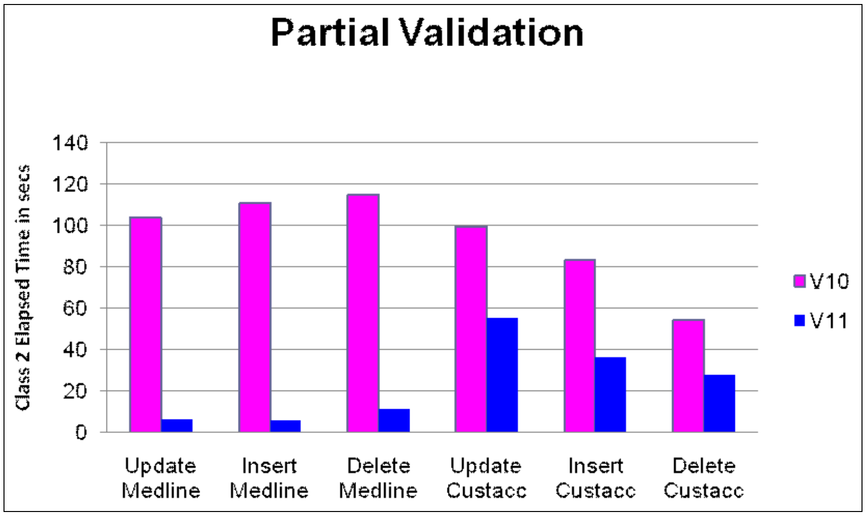

Figure 8-24 Partial validation: class 2 elapsed time

Figure 8-25 shows that, depending on the size of the XML document and the updated portion, there is a CPU reduction of 80 - 92% for Update, 63 - 85% for Delete, and 69 - 92% for Insert documents with partial validation.

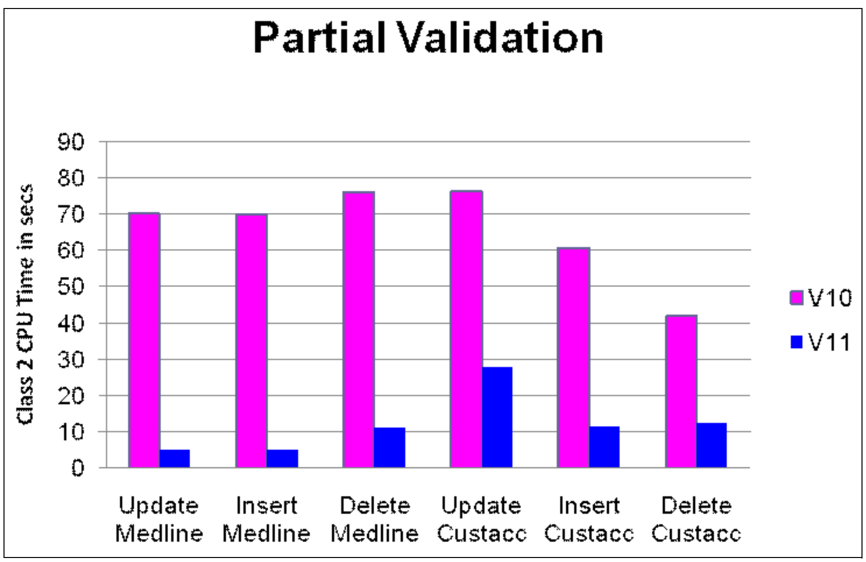

Figure 8-25 Partial validation: class 2 CPU time

The performance benefit varies with the size of the document and the size of the updated portion. For documents that are 10 MB in size, we observe up to a 92% CPU reduction and 93% elapsed time reduction. For smaller documents, about only 10 KB in size, we observe up to an 80% CPU reduction and 58% elapsed time reduction. Users with large XML documents can expect good savings with the new partial validation feature, as it is possible that large pieces of their documents no longer need to be revalidated.

The partial validation feature is available in DB2 11 and DB2 10 with APAR PM69178 applied.

8.5 Archive transparency

DB2 10 introduced system-period temporal table (STT), application-period temporal table (ATT), and bi-temporal table (BTT) with both STT and ATT features combined. To query historical data contained in the history table of STT or ATT, the table-reference of the FROM clause was extended by allowing the following timestamp expression clause after the table name:

- FOR SYSTEM_TIME (BUSINESS_TIME) AS OF timestamp-expression
- FOR SYSTEM_TIME (BUSINESS_TIME) FROM timestamp-expression1 TO time-stamp-expression2
- FOR SYSTEM_TIME (BUSINESS_TIME) BETWEEN timestamp-expression1 AND timestamp-expression2

However, this approach requires changing the underneath SQL statements by hardcoding period specifications in the application.

In DB2 11 for z/OS, two new special registers, CURRENT TEMPORAL SYSTEM_TIME and CURRENT TEMPORAL BUSINESS_TIME, are introduced to allow access to historical data at a particular point without laborious SQL changes.

When the special registers are set, DB2 adds the implicit period specification FOR SYSTEM_TIME (BUSINESS_TIME) AS OF CURRENT TEMPORAL SYSTEM_TIME (BUSINESS_TIME) to the static and dynamic SQL statements against STT, ATT, or BTT. Setting the special registers has no effect on SQL statements against regular non-temporal tables.

Both STT and ATT tables are required to have time period columns specified. This requirement has created an inconvenience for users of regular tables to migrate to STT or ATT. DB2 11 introduced the concept of archive-enabled table and archive table to pair up a current operational table and a history table. Unlike STT and ATT, these tables do not require time period columns, and the archive relationship between the archive-enabled table and the archive table is established through an ALTER TABLE … ENABLE ARCHIVE USE … statement.

DB2 11 for z/OS introduced two global variables to enhance the usability and archive transparency of these new types of tables:

- SYSIBMADM.MOVE_TO_ARCHIVE controls whether DB2 inserts rows that are deleted from this table into the associated archive table.
- SYSIBMADM.GET_ARCHIVE controls whether queries include or exclude archive table data without having to update the SQL.

8.5.1 Performance Impact of CURRENT TEMPORAL SYSTEM_TIME

The DB2 11 special register TEMPORAL SYSTEM_TIME only changes the way on how SYSTEM_TIME periods are specified for the AS OF scenario. If you specify a range for periods using FROM … TO… or BETWEEN…AND… clauses, you must continue to append the period specification in the FROM clause as you would in DB2 10. Additionally, special registers do not change how DB2 handles the query internally.

For example: Assume that, in DB2 10, we have a system-period temporal table called STT1 and its corresponding history table called STT1_HIST. In this scenario, consider the query in Example 8-5.

Example 8-5 Temporal query example

```
SELECT *
FROM STT1
FOR SYSTEM_TIME AS OF '2013-12-01-12.00.00.000000000'
WHERE SST1.ID=123
```

This query is rewritten by DB2 as shown in Example 8-6.

Example 8-6 Rewritten temporal query

```
SELECT *
FROM (SELECT * FROM STT1
WHERE SYS_START <= '2013-12-01-12.00.00.000000000'
AND SYS_END > '2013-12-01-12.00.00.000000000'
AND STT1.ID=123
UNION ALL
SELECT * FROM STT1_HIST
WHERE SYS_START <= '2013-12-01-12.00.00.000000000'
AND SYS_END > '2013-12-01-12.00.00.000000000' AND STT1_HIST.ID=123)
```

The query rewrite combines both STT1 and STT1_HIST through a UNION ALL operator, and it is done implicitly. The modified query performs the same as an identical query statement with UNION ALL hand-coded by the application developer. These queries are subject to the same DB2 optimizer rules, which define the access path for the UNION ALL operation.

In DB2 11, the following explicit specification of SYSTEM_TIME period in the query continues to work as before, as shown in Example 8-7.

Example 8-7 Temporal query in DB2 11

```
SELECT *
FROM STT1
FOR SYSTEM_TIME AS OF '2013-12-01-12.00.00.000000000'
WHERE SST1.ID=123
```

The two statements in Example 8-8 also work and lead to the same query rewrite if the SYSTIMESENSITIVE option is set to YES.

Example 8-8 DB2 11 temporal query rewrite

```
SET CURRENT TEMPORAL SYSTEM_TIME =
'2013-12-01-12.00.00.000000000';

SELECT * FROM STT1 WHERE SST1.ID = 123;
```

Only the original statement **`SELECT * FROM STT1 WHERE SST1.ID=123`** gets cached in the dynamic statement cache instead of the rewritten statement. There is an indicator in the EXPANSION_REASON column of the DSN_STATEMENT_CACHE_TABLE to inform the user of the implicit transformation.

The benefit of using CURRENT TEMPORAL SYSTEM_TIME special register is to void the need for SQL change. There is a bind time overhead for the query transformation, but there is no runtime performance difference between the implicitly rewritten query as a result of setting the special register and the explicit query combining two tables using UNION ALL.

8.5.2 Performance impact of CURRENT TEMPORAL BUSINESS_TIME

Like CURRENT TEMPORAL SYSTEM_TIME special register, CURRENT TEMPORAL BUSINESS_TIME also only changes the way how BUSINESS_TIME periods are specified for the AS OF scenario. If you specify a range for the periods using FROM … TO… or BETWEEN…AND… clause, you need to append the period specification in the FROM clause as in DB2 10.

The performance characteristics of setting CURRENT TEMPORAL BUSINESS_TIME are very similar to that of setting CURRENT TEMPORAL SYSTEM_TIME. Consider the queries in Example 8-9.

Example 8-9 Temporal query example

```
SET CURRENT TEMPORAL BUSINESS_TIME =
'2013-12-01-12.00.00.000000000';
UPDATE ATT1 SET c1 = c1 * 1.2 WHERE id = 123;
```

These two statements, combined with the BUSTIMESENSITIVE YES bind option, result in an implicit transformation and the update statement is executed as shown in Example 8-10.

Example 8-10 DB2 11 temporal query transformation

```
UPDATE ATT1 SET C1 = C1 * 1.1
WHERE bus_start <= '2013-12-01-12.00.00.000000000'
AND bus_end > '2013-12-01-12.00.00.000000000'
AND id = 123;
```

The transformation has the same effect as the statement with explicit business time specification as in DB2 10 shown in Example 8-11.

Example 8-11 DB2 10 temporal query with explicit business time specification

```
UPDATE ATT1 FOR BUSINESS_TIME
AS OF '2013-12-01-12.00.00.000000000'
SET c1 = c1 * 1.2 WHERE id = 123;
```

There is a bind time overhead for the query transformation, but there is no runtime performance difference between the implicitly rewritten query as a result of setting the special register and the explicit query.

8.5.3 Performance impact of SYSIBMADM.MOVE_TO_ARCHIVE global variable

DB2 11 introduces the SYSIBMADM.MOVE_TO_ARCHIVE global variable. It allows to periodically move data from archive-enabled tables to archive tables to reduce the size of archive-enabled tables. This helps improve the efficiency and performance of applications.

If the SYSIBMADM.MOVE_TO_ARCHIVE global variable is set to Y, historical data is stored in the associated archive table when a row is deleted in an archive-enabled table. The storing of a row of historical data in the archive table is "transparent" to the application. When the global variable is set to Y, an update operation returns an error.

We compared the performance of inserting rows into an archive table by setting this global variable and doing a delete against the archive-enabled table versus using delete triggers to move data to the archive table. This last approach is likely to be the application solution that most users may use to achieve the goal of moving deleted rows to an archive table before DB2 11 for z/OS.

Scenario 1: Setting SYSIBM.MOVE_TO_ARCHIVE global variable

Example 8-12 shows the DDL used to set up the archive-enabled table and archive table.

Example 8-12 DDL to set up archive-enabled table and archive table

```
CREATE TABLE WRK.WRKTB01 (FKEY SMALLINT          NOT NULL,
                          FSMALLINT SMALLINT     NOT NULL,
                          FINT INT               NOT NULL,
                          FFLOAT REAL            NOT NULL,
                          FDOUBLE DOUBLE         NOT NULL,
                          FBOOLEAN SMALLINT      NOT NULL,
                          FDATE DATE             NOT NULL,
                          FCHAR CHAR(6)          NOT NULL,
                          FVARCHAR VARCHAR(10)   NOT NULL,
                          FTIME TIME             NOT NULL,
                          FTIMESTAMP TIMESTAMP   NOT NULL,
                          FNUM DECIMAL(9,2)      NOT NULL)
   IN WRKDB01.WRKTS01;

CREATE TYPE 2 UNIQUE INDEX WRK.WRKIX01 ON
   WRK.WRKTB01 (FKEY ASC)
   USING STOGROUP WRKSG01
   PRIQTY 1000
   SECQTY 500
   FREEPAGE 0
   PCTFREE  10
   CLUSTER
   BUFFERPOOL BP0
   CLOSE NO;

CREATE TABLE WRK.WRKTB02 LIKE WRK.WRKTB01;

ALTER TABLE WRK.WRKTB01 ENABLE ARCHIVE USE WRK.WRKTB02;
```

The archive-enabled table WRK.WRKTB01 is loaded with 1 million rows. All 1 million rows were deleted after setting the global variable to “Y”, as shown in Example 8-13.

Example 8-13 Setting SYSIBMADM.MOVE_TO_ARCHIVE to Y

```
SET SYSIBMADM.MOVE_TO_ARCHIVE = 'Y';

DELETE FROM WRK.WRKTB01;
```

Scenario 2: Using delete trigger to move deleted rows to archive table

In this scenario, we turned off the archive relationship between WRK.WRKTB01 and WRK.WRKTB02 by executing the SQL shown in Example 8-14.

Example 8-14 Disabling archiving

```
ALTER TABLE WRK.WRKTB01 DISABLE ARCHIVE;
```

Then, we loaded 1 million rows back into WRK.WRKTB01. A delete trigger was defined as shown in Example 8-15.

Example 8-15 Creating a delete trigger

```
CREATE TRIGGER TRIG01
 AFTER DELETE ON WRK.WRKTB01
 REFERENCING OLD AS OLD_ROW
```

```
 FOR EACH ROW MODE DB2SQL
 BEGIN ATOMIC
   INSERT INTO WRK.WRKTB02
(FKEY, FSMALLINT, FINT, FFLOAT, FDOUBLE,
FBOOLEAN, FDATE, FCHAR, FVARCHAR, FTIME,
FTIMESTAMP, FNUM) VALUES
(OLD_ROW.FKEY, OLD_ROW.FSMALLINT, OLD_ROW.FINT, OLD_ROW.FFLOAT, OLD_ROW.FDOUBLE,
OLD_ROW.FBOOLEAN, OLD_ROW.FDATE,
OLD_ROW.FCHAR, OLD_ROW.FVARCHAR, OLD_ROW.FTIME, OLD_ROW.FTIMESTAMP, OLD_ROW.FNUM);
 END
```

Then, we deleted all rows from WRK.WRKTB01 and let the trigger move deleted rows to WRK.WRKTB02.

Figure 8-26 shows the performance comparison between scenario 1 and scenario 2. The time is in seconds. Setting the global variable SYSIBMADM.MOVE_TO_ARCHIVE outperforms the delete trigger by a large margin. This means that DB2 supplied archiving is much more efficient than user-defined triggers for this scenario.

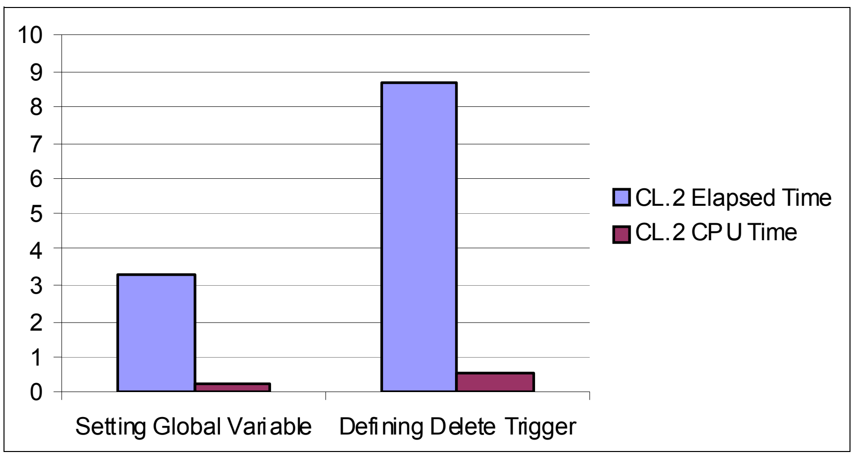

Figure 8-26 Archiving using global variable SYSIBM.MOVE_TO_ARCHIVE versus using delete trigger

8.5.4 Performance impact of SYSIBMADM.GET_ARCHIVE global variable

If the global variable SYSIBMADM.GET_ARCHIVE is set to “Y”, data is retrieved from the archive table when an archive-enabled table is referenced. In other words, SELECT … FROM an archive_enabled_table gets transformed into (SELECT … FROM archive_enabled_table) UNION ALL (SELECT … FROM archive_table).

This variable only impacts SELECT statements with corresponding archive tables defined. No transformation is done to INSERT, UPDATE, DELETE, MERGE operations and they do not include rows from any of the associated archive tables via UNION ALL.

Setting the SYSIBMADM.GET_ARCHIVE global variable causes bind time performance degradation, and the bind time overhead is measured to be ranging 10 - 80% of CPU and elapsed time increase depending on the complexity of the transformation. However, since the cost incurs infrequently in most cases, this overhead is considered acceptable given the benefit of not having to rewrite the query to obtain data from the archived table.

8.6 NoSQL JSON support

Rapidly changing application environments require a flexible mechanism to store and communicate data between different application tiers for the Cloud, Mobile, Social, and Big Data Analytics. The popularity of JavaScript for many of these new style applications makes JavaScript Object Notation (JSON) an attractive choice. JSON has proven to be a key technology for mobile, interactive applications by reducing overhead for schema designs and eliminating the need for data transformations. It is a lightweight data-interchange format, easy for humans to read and write, and easy for machines to parse and generate.

DB2 10 and 11 provide JSON support by providing server-side built-in functionality for storing, indexing, and retrieving JSON documents and server-side UDFs for JSON document access.

The functionality is not included in the base DB2 product. You need the no-charge feature Accessories Suite 3.1 (program number 5697-Q04) available as of 2013/12/06 for DB2 10.

IBM DB2 Accessories Suite for z/OS, V3.1 consists of three features, each bundling components designed to enhance your use of IBM DB2 for z/OS. This release includes the addition of and changes to the following components:

- Addition of IBM DB2 Accessories Suite for z/OS feature, in support for DB2 11 for z/OS, which contains new spatial functions
- Addition of JSON capability, which bundles necessary components that enable DB2 for z/OS to be used as a JSON document store
- Update to Data Studio 4.1, delivering health monitoring, single query tuning, and application development tools for DB2 for z/OS

Users can interact with JSON data in the following ways:

- They can administer and interactively query JSON data using a command-line shell.
- They can programmatically store and query data from Java programs using a driver for JSON supplied by IBM that enables them to connect to their JSON data through the same JDBC driver used for SQL access.

JSON data is stored as Binary JSON (BSON) in an inline binary large object (BLOB) in DB2. The inline BLOB is chosen for performance since most of JSON documents are small.

The JSON_VAL built-in function provides a SQL interface to extract and retrieve JSON data into SQL types from BSON objects. The JSON_VAL function assumes that each BLOB holds one JSON document stored as BSON, and that each such JSON document holds only one business object in JSON notation. In addition to JSON_VAL built-in function, Index on Expression (IoE) is extended to support BLOB input types. This extension is needed for performance to support index on JSON data since the JSON data is stored as BLOBs. The key value may come from the inline or outline portion of the inline BLOB. The user creates an IoE index over the JSON_VAL function with the BLOB containing the JSON data as input.

Since JSON data is stored as BLOB in a large object (LOB) table space, there are some restrictions on JSON index, which is a LOB expression-based index type:

- LOAD utility on a table with one or more JSON indexes defined
- CHECK DATA utility on a table with one or more JSON indexes defined on it
- REORG TABLESPACE SHRLEVEL CHANGE on a table space with one or more JSON indexes defined in it
- REBUILD INDEX SHRLEVEL CHANGE on a JSON index
- CHECK INDEX SHRLEVEL CHANGE on a JSON index

Besides the built-in function JSON_VAL, DB2 also includes a set of UDFs for storing and retrieving BSON documents in the DB2 BLOBS. These UDFs are clients that are available with DB2 for Linux, UNIX, and Windows 10.5, to manipulate JSON data stored in DB2 for z/OS.

The basic support for JSON in DB2 for z/OS is illustrated in Figure 8-27.

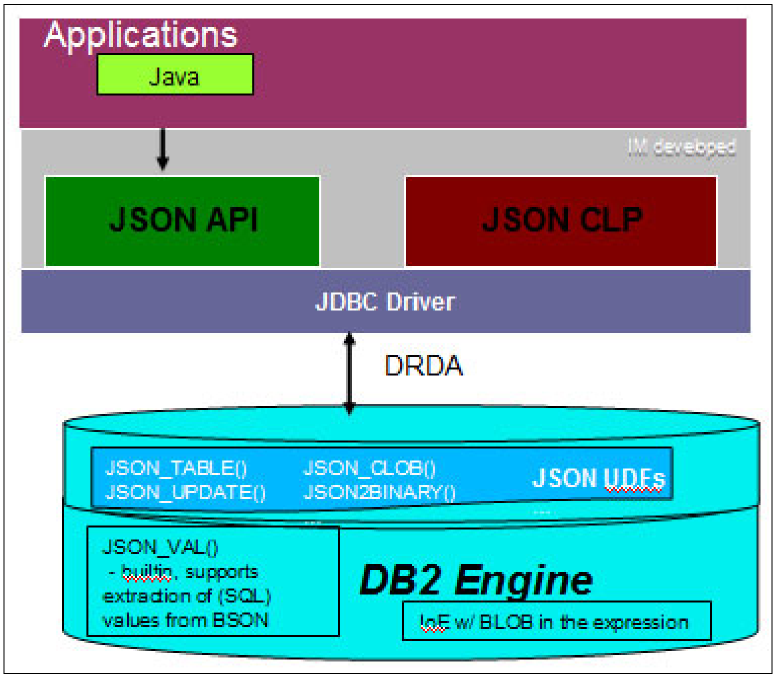

Figure 8-27 DB2 for z/OS and JSON

JSON support includes:

- A programming interface for Java applications to store, update, and intelligently query JSON documents
- A command line processor for performing administration tasks and data access operations on JSON data
- A Wire Listener service that extends the support to other languages via an open source wire protocol for accessing JSON data.

The JSON capability is a driver-based solution that embraces the flexibility of the JSON data representation within the context of an RDBMS that includes well-known enterprise features and quality of service.

With this offering, applications can manage JSON documents in DB2 for z/OS using an application programming interface (API) which is designed after the MongoDB data model and query language. This API leverages available DB2 for z/OS capabilities to store, modify, and retrieve JSON documents. This allows existing DB2 administration skills, resources, and processes to be used for managing this new type of data in DB2 for z/OS.

8.6.1 Built-in function JSON_VAL

DB2 for z/OS offers a new built-in function, JSON_VAL, that provides an SQL interface to extract and retrieve JSON data into SQL data types from BSON objects. The first argument to the JSON_VAL function is the BLOB column that contains one JSON document represented as BSON, and each such JSON document holds only one business object in JSON notation. The JSON_VAL built-in function returns an element of a JSON document identified by the JSON field name specified in the search string, and the value if the JSON element is returned in the data type and length specified in the result-type.

Assume that table CUSTOMERS is designed to store JSON data, and we want to find out how many customers have a postal code of '95123', then the following sample query could be used:

```
SELECT count(*) from MYSCHEMA.CUSTOMERS
  WHERE JSON_VAL(DATA, 'address.zipcode', 's:5') = '95123';
```

8.6.2 UDFs for JSON support

DB2 for z/OS also provides several user-defined functions (UDFs) under schema SYSTOOLS to support operations on JSON data. These UDFs are typically called by the JSON API to update or retrieve JSON data. Following are some of the important UDFs:

- SYSTOOLS.JSON_BINARY (INJSON BLOB (16M), INELEM VARCHAR (2048), MAXLENGTH INTEGER) RETURNS VARCHAR (2050) FOR BIT DATA

  This user-defined function returns a binary portion of a JSON data and is typically used in select projections or where clause binary comparisons.

- SYSTOOLS.JSON_TABLE (INJSON BLOB (16M), INELEM VARCHAR (2048), RETTYPE VARCHAR (100)) RETURNS TABLE (TYPE INTEGER, VALUE VARCHAR (2048))

  This function returns an array of elements in JSON data.

- SYSTOOLS.JSON_UPDATE (INJSON BLOB (16M), INELEM VARCHAR (32672)) RETURNS BLOB (16M)

  This function is used to update JSON data and usually called from JSON API.

- SYSTOOLS.JSON_LEN (INJSON BLOB (16M), INELEM VARCHAR (2048)) RETURNS INTEGER.

  This function returns the size of array of elements in JSON data, and returns NULL if an element is not an array.

8.6.3 Allocation of table spaces for JSON

When we use the JSON API to insert a JSON document into a collection, if the table that represents the collection to store the JSON document does not exist, DB2 implicitly creates a database and a table space before creating that table, and inserts a row into table SYSTOOLS.SYSJSON_INDEX.

In this case, the default buffer pool is used as the buffer pool for the table space; this way might not be a good method for achieving good performance.

Another way is to use explicit DDL to create a base table space and a LOB table space for the collection. However, if you use JCL to create tables to store JSON data, you need to create proper indexes for JSON data by using NOSQL command-line interface or JSON API. Example 8-16 on page 197 shows the DDL statements to create the table spaces for collection “BOOKS”, assigning the base table space to BP32K3 and the LOB table space to BP32K4.

Example 8-16 Sample DDL for explicit definition of base and LOB table spaces

```
CREATE TABLESPACE TSBOOKS IN JSONDB USING STOGROUP STGJSON FREEPAGE 0
   PCTFREE 0 MAXPARTITIONS 10 LOCKSIZE PAGE BUFFERPOOL BP32K3 CCSID UNICODE;
COMMIT;
CREATE LOB TABLESPACE TLBOOKS IN JSONDB USING STOGROUP STGJSON
    PRIQTY 10000 SECQTY 10000 BUFFERPOOL BP32K4 LOG NO;
COMMIT;
CREATE TABLE BOOKS    (
    ID CHAR (12) FOR BIT DATA NOT NULL,
    DATA BLOB(16777216) INLINE LENGTH 1024
    ) IN JSONDB.TSBOOKS;
```

Here is a sample command to create an index using the NoSQL command-line interface (NoSQL cmd):

```
nosql>db.BOOKS.ensureIndex({"ISBN": [1, "$string", 20]})
Index <BOOKS_xISBN> was created successfully.
```

With the execution of the *ensureIndex* to create a unique index on column ISBN with a length of 20 characters, the NOSQL command-line interface inserts a metadata row into table SYSTOOLS.SYSJSON_INDEX. See Figure 8-28.

SYSTOOLS.SYSJSON_INDEX

ID [INTEGER]	IDXNAME [VARCHAR(128)]	COLLECTION [VARCHAR(255)]	FIELD [VARCHAR(255)]	DATATYPE [VARCHAR(30)]
162		CITYLOTS	ID	CHAR(12) FOR BIT DATA
241	BOOKS_xISBN	BOOKS	ISBN	VARCHAR(20)

Figure 8-28 Metadata for index

And inserts a row into SYSIBM.SYSINDEXES. See Figure 8-29.

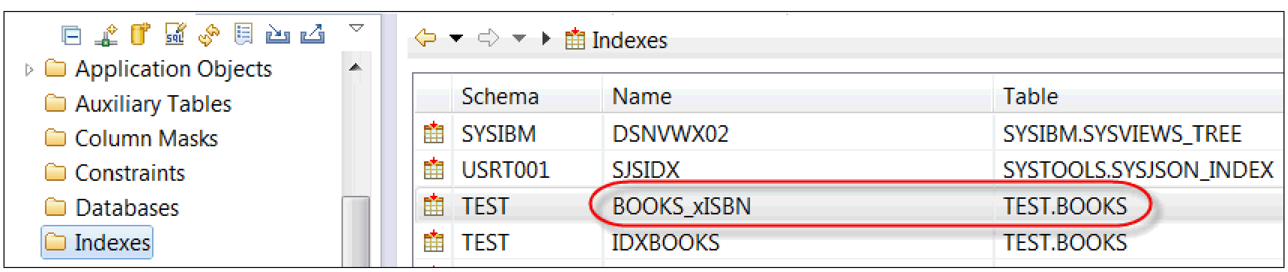

Figure 8-29 Entry in SYSINDEXES

Note the following considerations:

- The base table space needs to be "CCSID UNICODE"
- Inline length could be used for performance purposes. However, we need to estimate the best size for inline length. The average size of the LOB column could be used as the best estimate inline length.
- The JSON data is stored as BSON in column DATA.

8.6.4 JSON performance

We examine the performance for INSERT, SELECT, UPDATE, and DELETE statements on JSON documents.

These measurements are preliminary and have been done with the JSON functionality available with the Accessories Suite for DB2 10. Some results are still under investigation at the time of writing.

INSERT performance

To store a JSON document into a DB2 table, users could write a Java application using the JSON Java API that comes with the DB2 Accessories Suite or the users could directly insert the BLOB into column DATA, which is used to store the JSON document. If a BLOB data is inserted into column DATA directly, data must be a well-formed JSON document. Inserting JSON data into DB2 by using JSON Java API can be done with some simple steps:

1. Get the DB object:

   ```
   DB db = NoSQLClient.getDB (conn);
   ```

2. Get a collection handle:

   ```
   DBCollection col = db.getCollection ("BOOKS");
   ```

3. Insert a document:

   ```
   BasicDBObject json = new BasicDBObject ();
   json.append ("AUTHOR", author);
   json.append("TITLE", title);
   json.append("ISBN", isbn);
   ison.append("PRICE", price);
   …..
   coll.insert (json);
   ```

Table 8-1 shows the measurements of inserting a JSON document API into a BLOB column using the JSON Java. Inserting a JSON document using JSON Java API takes a bit longer than using SQL because the client needs to construct a well-formed JSON document before inserting it.

Table 8-1 Inserting a JSON document

Time in microseconds	CL1 elapsed time	CL2 elapsed time	CL1 CP time	CL2 CP time
SQL	607	557	134	101
API	636	557	146	101

SELECT performance

With the new built-in function JSON_VAL, users are able to retrieve JSON data from the BLOB column DATA directly. This function is an interface that the JSON Java API uses to extract and retrieve JSON data into SQL data types from BSON objects. The JSON Java API provides method db.collection.find(), similar to the find() method from MongoDB. To evaluate the performance of SELECT, a query that selects 2 JSON columns from JSON data, was compared against the similar operation using JSON Java API, which has 2 JSON columns in the projection.

This example shows the statement that was performed to assess the SELECT with built-in function JSON_VAL performance. Here is an example of JSON_VAL usage:

```
String str = " SELECT JSON_VAL(DATA, \'properties.BLOCK_NUM\', \'s:10\'), " +
   "JSON_VAL(DATA, \'properties.LOT_NUM\', \'s:10\') " +
    " FROM  TEST.CITYLOTS WHERE JSON_VAL(DATA, \'properties.BLKLOT\', \'s:10:na\')
   ‘=  " ;
```

Table 8-2 shows the elapsed time and CPU time in microseconds, of elapsed and CPU time of the SELECT statement and JSON Java API. The API costs more than selecting directly from the table does. This means that using the API, which in turn invokes the built-in function JSON_VAL, has some overhead. However, the MongoDB like API provides a rich set of operations to search and retrieve JSON documents.

Table 8-2 Selecting a JSON document

Time in microseconds	CL1 elapsed time	CL2 elapsed time	CL1 CP time	CL2 CP time
SELECT	138	83	115	82
find()	201	87	141	86

UPDATE performance

To update data in a JSON document, we could use the SQL UPDATE statement. However, with this method, the entire value of column DATA is updated and the input to the JSON data must be a well-formed JSON document. Alternatively, DB2 JSON API method DBCollection.update() can be used to update specific fields in the document. Specific fields in the document can be updated or appended, or the entire document can be replaced depending on the update parameters. Another method such as DBCollection.FindAndModify() can be used to find a document, modify some specific fields, then return that document.

Table 8-3 presents the elapsed time and CPU time, in microseconds, of updating a JSON column using an UPDATE statement and JSON Java API FindAndModify(). Using the API costs more than directly updating a BLOB column in the table using SQL because FindAndModify() needs to invoke several UDFs to find the position of the column to be updated, and then uses UDF update() to update data and return the updated data.

Table 8-3 Updating a JSON document

Time in microseconds	CL1 elapsed time	CL2 elapsed time	CL1 CP time	CL2 CP time
UPDATE (SQL)	324	170	197	158
FindAndModify()	674	432	257	197

DELETE performance

To delete data in a JSON document, we could use the SQL DELETE statement. In addition, there are several DB2 JSON API methods to delete one or more JSON documents. DBCollection.remove() can be used to delete all documents or just delete specific documents in a collection. Another method such as FindAndRemove() can be used to delete a document in a collection; this method also returns the removed document.

Table 8-4 presents the elapsed time and CPU time, in microseconds, of deleting a JSON document by using the DELETE statement and JSON Java API. Data shows that using the API costs more than directly deleting a BLOB column in the table. However, using the JSON API is easier to program for Java script.

Table 8-4 Deleting a JSON document

Time in microseconds	CL1 elapsed time	CL2 elapsed time	CL1 CP time	CL2 CP time
DELETE	382	333	125	92

Time in microseconds	CL1 elapsed time	CL2 elapsed time	CL1 CP time	CL2 CP time
remove()	532	471	159	116
FindAndRemove	496	379	205	145

Besides the CLP and a rich set of Java JSON APIs, similar to the MongoDB JSON API for Java, that DB2 for z/OS Accessories Suite offers, DB2 provides the built-in function JSON_VAL and several UDFs for storing and retrieving Binary JSON (BSON) documents in the DB2 BLOBS columns.

8.7 Global variables

Traditionally, within a relational database system, most interactions between an application and the DBMS are in the form of SQL statements within a connection. In order to share information between SQL statements within the same application context, the application that issued the SQL statements would have to do this work by copying the values from the output arguments, such as host variables, of one statement to the input host variables of another. Similarly, when applications issue host-language calls to another application, host variables need to be passed among applications as input or output parameters in order for the applications to share common variables. Furthermore, SQL statements defined and contained within the DBMS, such as the SQL statements in the trigger bodies, could not access this shared information at all.

These restrictions limit the flexibility of relational database systems and thus, the ability of users of such systems to implement complex, interactive models within the database itself. Users of such systems are forced to put supporting logic inside their applications in order to access and transfer user application information and database information within a relational database system. Ensuring the security of the information being transferred and accessed is also left to the user to enforce in their application logic.

To overcome this restriction and to maximize the flexibility of a DBMS, global variables are introduced in DB2 11 for z/OS. A global variable can be created, instantiated, accessed, and modified by the applications. Global variables are named memory variables that you can access and modify through SQL statements. Global variables enable you to share relational data between SQL statements without the need for application logic to support this data transfer. You can control access to global variables through the GRANT (global variable privileges) and REVOKE (global variable privileges) statements.

With the introduction of global variables, users can now freely build a complex system within a relational database where information can be shared between SQL statements on the same sessions, or can be accessed by SQL statements defined and contained within the database system.

A global variable is associated with a specific application context, and contains a value that is unique to that application scope. A created global variable is available to any active SQL statement running against the database on which the variable was defined. A global variable can be associated with more than one application scope, but its value is specific to each application scope. SQL statements sharing the same connection (that is, under the same application scope) can create, access, and modify the same global variables. Another important aspect of global variables is that when security of the information is an issue, controlling access is possible.

Specific privileges are required to create or drop a global variable and to read or modify the contents of one. This rule also applies to their definition: privileges associated with a global variable are also defined within the system catalog. This enhancement includes the following functional additions to DB2.

8.7.1 DDL and catalog information

A new DDL statement allows the application to create a global variable to be shared among SQL statements using the same connection. The CREATE VARIABLE statement creates a new variable object in DB2, and the definition is saved in the DB2 catalog. The definition is shared.

When the variable gets instantiated, if the DEFAULT clause is not specified, NULL is used as the default value. The value of a global variable can be changed by SET, SELECT INTO, or VALUES INTO, or as an argument of an OUT or INOUT parameter in a CALL statement. The content of each global variable is not affected by COMMIT or ROLLBACK statements.

The SYSIBM.SYSVARIABLES table contains one row for each global variable that has been created. The SYSIBM.SYSVARIABLEAUTH table contains one row for each privilege of each authorization ID that has privileges on a global variable. The SYSIBM.SYSDEPENDENCIES table records the dependencies between objects.

A sample CREATE statement is depicted in Example 8-17.

Example 8-17 Sample create global variable statement

```
CREATE VARIABLE BATCH_START_TS TIMESTAMP
DEFAULT CURRENT TIMESTAMP;
```

The length attribute and data type of a global variable does not vary after it is created. Furthermore, you cannot ALTER the global variable definition at all, including its default value. When you no longer want the definition of the global variable to exist, you need to drop it. The same privileges are required to drop global variables as any other database object.

A sample DROP statement is depicted in Example 8-18.

Example 8-18 Sample DROP statement

```
DROP VARIABLE BATCH_START_TS;
```

8.7.2 Qualifying global variable

Global variable names are qualified two-part names. For unqualified global variables, the implicit qualifier is used to facilitate the naming resolution of global variables. DB2 determines the implicit qualifier for global variables as follows.

The schemas in the SQL PATH are searched in order from left to right for a matching global variable. If a global variable matches the global variable name in reference, resolution is complete. If no matching global variable is found after completing this step, an error is returned.

8.7.3 Global variable's scope

The scope of global variable's definition is similar to that of DB2 special register's, in that, when created, the definitions of global variables are shared across different DB2 connections.

However, each connection maintains its own instance of the global variable, such that the variable's content is only shared among SQL statements within the same connection.

For instance, if you use a global variable, which was created using the DDL in Example 8-17 on page 201 in a program, the first invocation of this global variable has the same value as the current time stamp. The subsequent use of this global variable would retain the initial instantiated value during the connection.

Example 8-19 shows how a global variable's value of 2013-08-02-14.59.46.423414 remains the same in different SQL statements (when referenced) within the same DB2 connection.

Example 8-19 Initial execution of the SQL

```
SELECT BATCH_START_TS, CURRENT TIMESTAMP
FROM SYSIBM.SYSDUMMY1
```

The result set from the initial execution is shown in Example 8-20.

Example 8-20 Result of global variable after the first execution

```
BATCH_START_TS CURRENT TIMESTAMP
2013-08-02-14.59.46.423414 2013-08-02-14.59.46.423414
```

The second execution of the same SQL statement in the same SQL Processor Using File Input (SPUFI) session is shown in Example 8-21.

Example 8-21 Second execution of SQL

```
SELECT BATCH_START_TS, CURRENT TIMESTAMP
FROM SYSIBM.SYSDUMMY1
```

The result set from the second execution is shown in Example 8-22.

Example 8-22 Result set from the second execution

```
BATCH_START_TS CURRENT TIMESTAMP
2013-08-02-14.59.46.423414 2013-08-02-14.59.46.424678
```

The third execution of the same SQL statement in the same SPUFI session is shown in Example 8-23.

Example 8-23 The third SQL execution

```
SELECT BATCH_START_TS, CURRENT TIMESTAMP
FROM SYSIBM.SYSDUMMY1
```

The result set from the third execution is shown in Example 8-24.

Example 8-24 Result set from the third execution

```
BATCH_START_TS CURRENT TIMESTAMP
2013-08-02-14.59.46.423414 2013-08-02-14.59.46.4252
```

If you rerun the above set of SQL statements at a different point (for instance, in a different SPUFI session another time), it would result in a different instantiated value for the global variable and that value would remain in effect until the end of that connection.

8.7.4 Global variable's naming resolution

The DB2 naming resolution precedence rule is modified to include global variable references. For global variable names, DB2 searches the SQL path and selects the first schema in the path such that the global variable exists in the schema and the user has authorization to use the global variable.

If at the time of naming resolution, the definition of the referenced global variable does not exist, message DSNX200I or DSNX100I is issued during BIND if VALIDATE (BIND) or VALIDATE (RUN) were specified, respectively, on the **BIND** command.

> **Note:** It is also possible that during static bind time, the checked objects preceding global variables in naming resolution might not exist yet, resulting in the object name being resolved to global variables, provided the variable definitions exist. In this case, the resolved name remains as global variables, even if the object becomes available before the execution of this statement because the naming resolution had already been done at static bind time.

8.7.5 Transactions

If a global variable is created within a session, it cannot be used by other sessions until the unit of work has committed. However, the newly created variable can be used within the session that created it before the unit of work is committed. This behavior is consistent with other created objects such as tables.

The setting of a global variable's value is non-transactional. Hence, an application cannot roll back the setting of the value of a global variable. Note the following conditions:

- If the creation of a global variable is rolled back, the variable no longer exists.
- If the drop of a global variable is rolled back, the value of the variable is what was before the drop.

8.7.6 Using global variables

After a global variable is created, any session in the database can use it providing the right privileges are owned. The READ privilege is required to reference a global variable and the WRITE privilege is required to modify the value of a global variable. The GRANT statement allows an authorized user to grant these privileges and the REVOKE statement is used to remove them. The owner of the global variable is explicitly granted all privileges on the variable. Also, each session has its own private value of a given global variable. A global variable is instantiated to its default value when it is first referenced in the session.

A sample GRANT statement to a global variable is shown in Example 8-25.

Example 8-25 Sample grant global variable statement

```
GRANT READ ON VARIABLE MYSCHEMA.BATCH_START_TS  TO PUBLIC;
GRANT WRITE ON VARIABLE MYSCHEMA.BATCH_START_TS  TO SYSADM;
```

The value of a global variable can be changed using the SET, SELECT INTO, VALUES INTO statements. They can also be changed by being the argument of an out or inout parameter in a call statement.

An example to modify the value of the MYSCHEMA.BATCH_START_TS variable to current time stamp is shown in Example 8-26.

Example 8-26 Modify the value of a global variable statement

```
SET MYSCHEMA.BATCH_START_TS  = CURRENT TIMESTAMP;
```

Global variables can be used to calculate values in advance of their use and thus speed up performance and reduce complexity in other SQL entities. For example, a variable could be set by invoking a function that supplies the value of the SESSION_USER to fetch the department number for the current user. A view could use the value of this global variable in a predicate to select only those rows for which the user's department is assigned. The view does not need to contain the subquery. Therefore, it is less complex and performance for actions against the view is more efficient and faster:

1. Create the global variable that calls a UDF get_deptno:

   ```
   CREATE VARIABLE schema1.gv_deptno INTEGER
        DEFAULT(get_deptno (SESSION_USER));
   ```

   Or, instead of calling a function, you can use the fullselect in the variable creation definition:

   ```
   CREATE VARIABLE schema1.gv_deptno INTEGER DEFAULT
       ((SELECT deptno FROM hr.employees WHERE empUser = SESSION_USER));
   ```

2. Then, create a view:

   ```
   CREATE VIEW schema1.emp_filtered AS SELECT * FROM employee
         WHERE deptno = schema1.gv_deptno;
   ```

3. Modify permissions so other users can only SELECT from that view:

   ```
   REVOKE ALL ON schema1.emp_filtered FROM PUBLIC;
   GRANT SELECT ON schema1.emp_filtered TO PUBLIC;
   ```

8.7.7 Summary

With these new database objects, users can easily build complex systems where information can be shared between SQL statements without any application logic to support the transfer of information. When global variables are created, their definition is stored in the catalog tables and any session can use them. Each global variable has session scope, so every session has its own private value that it can manipulate and use and no other session can access that value. Another important aspect of global variables is that where security of the information is an issue, controlling access is possible.

8.8 Grouping sets, ROLLUP, and CUBE support

Online analytical processing (OLAP) and data warehouse application require the use of CUBE and ROLLUP functions to reach the analysis objective. CUBE and ROLLUP have proven to be frequently used functions in an OLAP database. DB2 11 for z/OS now provides CUBE and ROLLUP support in the SQL syntax.

Grouping-sets and super-groups are two new options under GROUP BY clause of the SELECT statement. A super-group stands for ROLLUP, CUBE clause. ROLLUP is helpful in providing subtotaling along a hierarchical dimension such as time or geography. CUBE is helpful in queries that aggregate based on columns from multiple dimensions.

With support for ROLLUP, CUBE, and grouping-sets specifications, the SQL coding complexity can be reduced greatly and the SQL performance can be improved dramatically. This support also provides DB2 family compatibility since DB2 for Linux, UNIX, and Windows already supports CUBE and ROLLUP.

The CUBE, ROLLUP, and GROUPING SETS function is effective in DB2 11 New Function Mode.

The GROUP BY clause specifies a result table that consists of a grouping of the rows of intermediate result table that is the result of the previous clause. The syntax diagram is shown in Figure 8-30.

In its simplest form, a GROUP BY clause contains a grouping-expression. More complex forms of the GROUP BY clause include grouping-sets and super-groups.

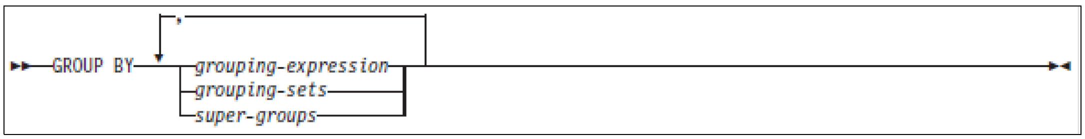

Figure 8-30 GROUP BY syntax diagram

The following sections illustrate the use of GROUPING SETS, ROLLUP, and CUBE clauses of the GROUP BY clause in subselect queries.

8.8.1 Grouping sets

Grouping sets can be thought of as the union of two or more groups of rows into a single result set. It is logically equivalent to the union of multiple subselects with the group by clause in each subselect corresponding to one grouping set.

Example 8-27 Grouping sets

```
CREATE TABLE TEST.T1(C1 INT, C2 INT, C3 INT, C4 INT)
INSERT INTO TEST.T1 VALUES (1, 2, 2, 4);
INSERT INTO TEST.T1 VALUES (2, 1, 3, 4);
INSERT INTO TEST.T1 VALUES (3, 3, 3, 4);
INSERT INTO TEST.T1 VALUES (4, 4, 3, 6);
--------------------------------------------------------------------------------
SELECT C3, C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY GROUPING SETS((C3), (C4))
is logically equivalent to
SELECT C3, NULL AS C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY C3
UNION ALL
SELECT NULL AS C3, C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY C4
```

Table 8-5 on page 206 shows the results of the grouping sets in Example 8-27.

Table 8-5 Results of the grouping sets

C3	C4	SUM_C1
2	NULL	1
3	NULL	9
NULL	4	6
NULL	6	4

8.8.2 ROLLUP

A ROLLUP grouping is an extension to the GROUP BY clause that produces a result set containing subtotal rows in addition to the “regular” grouped rows. Subtotal rows are “super-aggregate” rows that contain further aggregates whose values are derived by applying the same column functions that were used to obtain the grouped rows. These rows are called subtotal rows because that is their most common use; however, any column function can be used for the aggregation.

A ROLLUP grouping is a series of grouping-sets. The general specification of a ROLLUP with n elements:

```
GROUP BY ROLLUP(C1,C2,...,Cn-1,Cn)
```

is equivalent to

```
GROUP BY GROUPING SETS((C1,C2,...,Cn-1,Cn),
(C1,C2,...,Cn-1),
...
(C1,C2),
(C1),
() )
```

Example 8-28 shows a ROLLUP.

Example 8-28 ROLLUP example

```
SELECT C3, C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY ROLLUP(C3,C4)
is logically equivalent to
SELECT C3, NULL AS C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY GROUPING SETS((C3,C4),(C3),())
```

The results of the ROLLUP in Example 8-28 are shown in Table 8-6.

Table 8-6 Results of the ROLLUP

C3	C4	SUM_C1
2	4	1
3	4	3
3	6	4
2	NULL	1

C3	C4	SUM_C1
3	NULL	4
NULL	NULL4	4

8.8.3 CUBE

A CUBE grouping is an extension to the GROUP BY clause that produces a result set that contains all the rows of a ROLLUP aggregation and, in addition, contains “cross-tabulation” rows. Cross-tabulation rows are additional “super-aggregate” rows that are not part of an aggregation with subtotals.

Like a ROLLUP, a CUBE grouping can also be thought of as a series of grouping-sets. In the case of a CUBE, all permutations of the cubed grouping-expression-list are computed along with the grand total. Therefore, the elements of a CUBE translate to 2**n (2 to the power n) grouping-sets. For instance, a specification of:

```
GROUP BY CUBE(a,b,c)
```

is equivalent to

```
GROUP BY GROUPING SETS((a,b,c),
(a,b),
(a,c),
(b,c),
(a),
(b),
(c),
() )
```

Example 8-29 shows a CUBE.

Example 8-29 CUBE example

```
SELECT C3, C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY CUBE(C3,C4)
--is logically equivalent to--
SELECT C3, NULL AS C4, MAX(C1) AS MAX_C1
FROM TEST.T1
GROUP BY GROUPING SETS((C3,C4),(C3),(C4), ())
```

The results of the CUBE in Example 8-29 are shown in Table 8-7.

Table 8-7 Results of the CUBE

C3	C4	SUM_C1
2	4	1
3	4	3
3	6	4
NULL	4	3
NULL	6	4
2	NULL	1

C3	C4	SUM_C1
2	NULL	4
NULL	NULL	4

8.8.4 Performance

The CUBE and ROLLUP need many GROUP BY for the intermediate result before getting the final result set. Which means DB2 needs to sort on some columns that are based on the intermediate result. Example 8-30 shows a CUBE on three columns.

Example 8-30 Sort for CUBE

```
SELECT C1, C2, C3, max(C4)
FROM TEST.TAB
GROUP BY CUBE (C1, C2 , C3)
```

DB2 needs six sorts on different columns. When DB2 tries to get the final result, it needs to run GROUP BY on the following columns when building the intermediate results:

1. (C1,C2,C3)
2. (C1,C2)
3. (C2,C3)
4. (C1)
5. (C2)
6. (C3)

DB2 tries to reuse some of the group by intermediate results. In other words, DB2 can avoid running some of the sort. DB2 first sorts on #1(C1,C2,C3) columns group, then when it needs to sort for #2 columns group(C1,C2), it reuses the result of #1 to avoid sorting again on #2. Child #4(C1) also benefits with #1's sort result. Such behavior also happens for sort on #3(C2,C3) and sort avoidance on #5 (C2).

Figure 8-31 demonstrates the reduction by using sort avoidance.

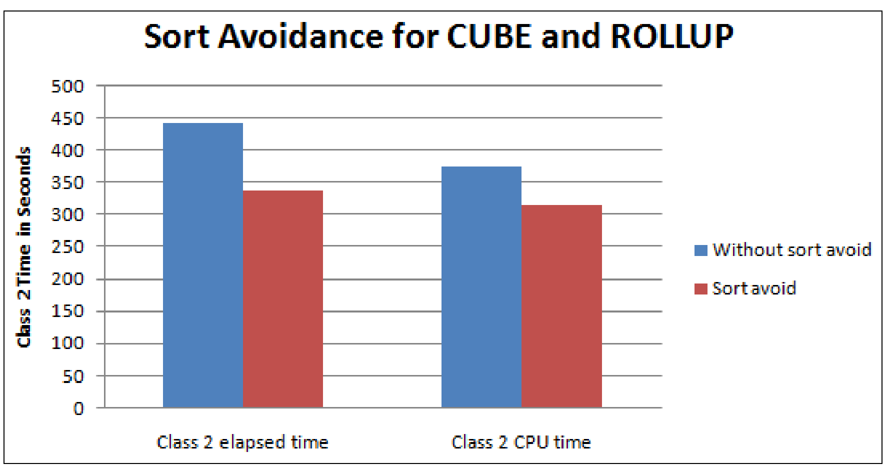

Figure 8-31 Sort avoidance for CUBE and ROLLUP

Lab measurements showed that CL2 CPU time is reduced by 15% for such sort avoidance in CUBE and ROLLUP function. To enable the sort avoidance, no user action is necessary.

8.8.5 Summary

With the new CUBE and ROLLUP SQL syntax, users can easily write OLAP queries. CUBE and ROLLUP, like other OLAP functions, need CPU time. There is a trade-off between application development requirement and DB2 CPU time.

9

Distributed environment

DB2 11 provides a number of enhancements to improve the availability and performance of distributed applications. DB2 11 also introduces further improvements of universal drivers for accessing data on any local or remote server.

This chapter describes these performance topics related to distributed access to DB2 for z/OS:

- Distributed access to DB2 11 for z/OS
- DDF Synchronous Receive support
- DDF package-based continuous block fetch
- Implicit commit for stored procedures
- Array data type performance
- Support for multi-threaded Java stored procedure environment
- Optimizations for calling stored procedures with local JDBC/ODBC drivers
- Distributed IRWW workload measurements

9.1 Distributed access to DB2 11 for z/OS

Distributed clients communicate to DB2 11 for z/OS using the IBM Distributed Relational Database Architecture (DRDA) protocol. DRDA is an open, vendor-independent architecture that supports the connectivity between a client and database servers. It was initially developed by IBM and then adopted by The Open Group as an industry standard interoperability protocol.

In DRDA terms, the Application Requester function accepts SQL requests from an application and redirects them to an Application Server for processing. The Application Server function receives requests from Application Requesters and processes them. The Application Server can process part of the request, and forwards the applicable portion of the request to a database server.

In a distributed application environment accessing DB2 for z/OS, the Application Requester function is supported by a DB2 for z/OS, a DB2 Client, a DB2 driver, or an IBM DB2 Connect™ server. The Application Server function is integrated in DB2 for z/OS.

Improvements related to the distributed access to DB2 11 to z/OS might require changes at the Client, Driver, or DB2 Connect side.

DB2 Clients, Drivers, and DB2 Connect DB2 10.5 FP2, and TCP/IP APAR PM80004 for z/OS 1.13, are required to fully use DB2 11 for z/OS distributed features, such as:

- CALL with array type arguments
- Larger CLIENT INFO properties (including new correlation token)
- Implicit COMMIT for stored procedures
- Sysplex support for global session variables

Note: At the time of this writing, the latest drivers and clients are available at the web page “Download initial DB2 10.5 clients and drivers” at the following link:

http://www.ibm.com/support/docview.wss?uid=swg21385217

z/OS APAR PM80004 enables new function to provide support for notification when a TCP connection is terminated.

The DRDA protocol implements DRDA levels to group improvements and features. An earlier DRDA Client works with DB2 11, but it may not use all the benefits of DB2 11. Any in-service level of DB2 Client, DB2 Drivers, or DB2 Connect server should work with DB2 11 conversion mode (CM) and DB2 11 new-function mode (NFM).

DB2 Connect drivers seamlessly handle the migration path from DB2 10 for z/OS to DB2 11 CM, and from them to DB2 11 NFM. In a data sharing environment, applications continue to function as members are migrated one by one.

9.2 DDF Synchronous Receive support

Before this DB2 11 improvement, DB2 server used asynchronous receive in DDF. This requires extra service request block (SRB) dispatching when data becomes available before processing the incoming request.

With z/OS 1.13 Communication Server and APAR PM80004 or z/OS 2.1, DB2 11 DDF server can set up a TCP/IP connection termination notification. DB2 11 replaces asynchronous calls by synchronous calls eliminating extra dispatches and additional overhead to complete a receive.

The new exploitation of the synchronous receive model allows DDF to be notified of any socket failure, even while DDF is in sync receive mode. This means that DDF can abort threads even when there is no outstanding communication request.

Long running DDF transactions with multiple statements with multiple send/receive processing at the server, will greatly benefit from this improvement. DDF transactions that mainly do FETCH processing with fewer receive processing at the server, or single message transactions, will not benefit from this improvement.

This new improvement does not require a rebind or any change in the application and it is applicable only to DB2 systems defined to use DDF inactive (subsystem parameter CMTSTAT=INACTIVE) thread processing.

Table 9-1 presents delta of normalized internal throughput rate (ITR), DIST CPU per commit, Total CPU per commit, CL1 CP and CL1 elapsed time of three non-stored procedure workloads of the distributed IRWW workload suite. Data shows that the most contribution to the reduction of Total CPU per commit is the reduction of DIST CPU per commit, which is the CPU spent by the distributed address space. The Total CPU per commit reduction varies depending on the workload. It varies 6 - 15%, and elapsed time reduction is about 2 - 9%, while normalized ITR improves 7 - 18%.

Table 9-1 DB2 10 versus 11 compared for distributed IRWW, non-stored procedure workload

PKGREL	**COMMIT**			**BNDOPT**		
Type	**SQCL**	**JDBC**	**SQLJ**	**SQCL**	**JDBC**	**SQLJ**
ITR	9.30	17.96	7.63	12.85	19.32	11.12
DIST CPU/ Commit	7.53	14.35	7.13	9.42	13.97	8.95
Total CPU/ Commit	7.10	14.26	6.51	10.67	15.21	9.10
CL1 CPU	13.70	9.11	14.21	10.28	10.19	21.56
CL1 elapsed	2.17	9.69	2.27	4.06	8.80	2.36

The distributed IRWW workload at 9.8, "Distributed IRWW workload measurements" on page 225 includes this one and other DRDA enhancements.

9.3 DDF package-based continuous block fetch

DB2 11 introduces package-based continuous block fetch. It can improve the performance of retrieval of large read-only result sets for a DB2 for z/OS application accessing a remote DB2 for z/OS server. Package-based continuous block fetch reduces significantly number of messages to be transmitted from the requester to retrieve the entire result set. Thus this new package-based continuous block fetch is more efficient than SQL-based continuous block fetch.

With package-based continuous block fetch, the requester opens a secondary connection to the DB2 server for each read-only cursor. The DB2 server returns extra query blocks until the cursor is exhausted or until the application closes the cursor. After the DB2 cursor is closed, DB2 closes the connection.

Figure 9-1 shows how SQL-based continuous block fetch works. Using this technique, DB2 can send numerous query blocks per request. A single connection is used for all SQL. The implication for the single connection is that other SQL, outside of the cursors, cannot use the connection while the cursor driven blocks are using the connection.

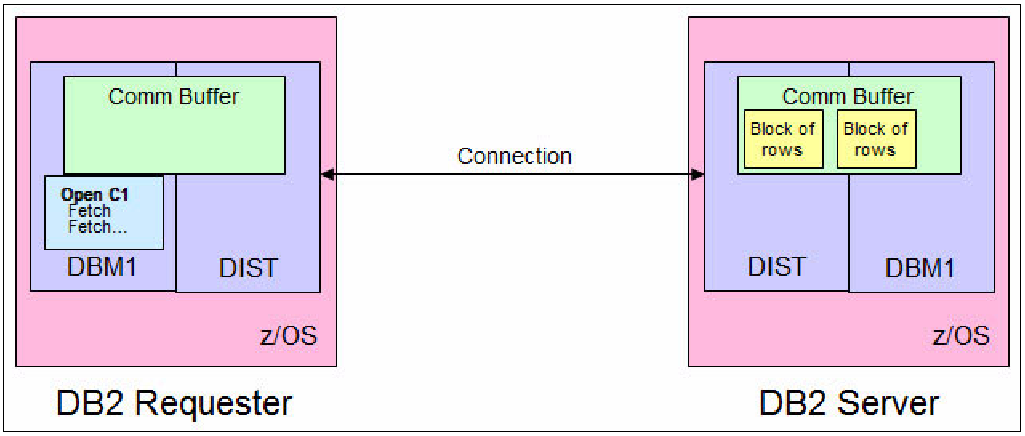

Figure 9-1 SQL-based continuous block fetch

Figure 9-2 shows a representation of the package-based continuous block fetch, introduced in DB2 11. With this method, query blocks flow on a secondary connection until the cursor is exhausted. As a consequence, the network latency is significantly improved. When the result set or cursor is exhausted, the DB2 server terminates the connection and the thread is immediately pooled.

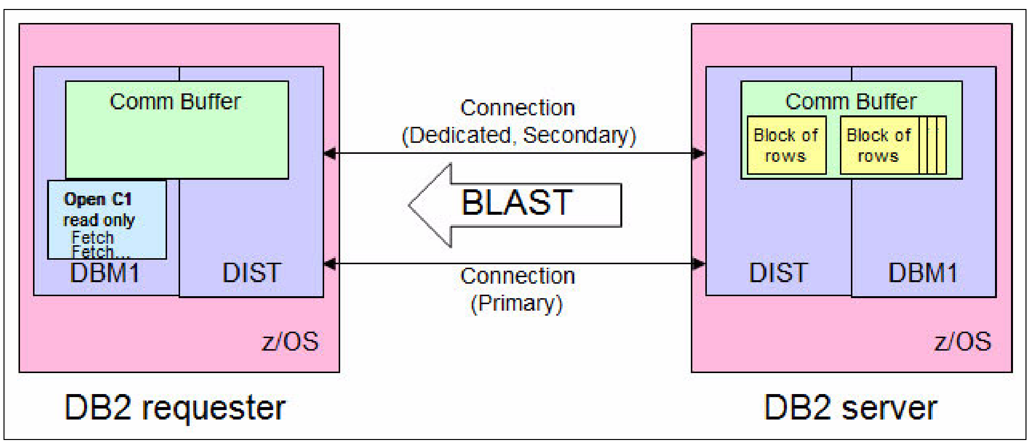

Figure 9-2 Package-based continuous block fetch

Package-based continuous block fetch provides a performance advantage for a DB2 for z/OS application with the following characteristics:

- The application queries only remote sites
- The application does not contain INSERT, UPDATE, DELETE, or MERGE statements
- No statement in the application creates a unit of recovery on the remote site. This situation results in an SQL error when the application package is bound for package-based continuous block fetch

Package-based continuous block fetch is easier to configure than the previously existing SQL-based continuous block fetch. You only need to bind the applications with the new DBPROTOCOL(DRDACBF) option. You do not need to modify your applications or set subsystem parameters to indicate the maximum number of blocks to be returned for a remote request.

Example 9-1 shows how to REBIND a package with the new option DBPROTOCOL(DRDACBF) on the requester.

Example 9-1 Option DBPROTOCOL(DRDACBF) at REBIND

```
//DRDACBF1 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB  DD DSN=DB1AT.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 DSN SYSTEM(DB1A)
 REBIND PACKAGE(DSN8BH11.DSN8BC3) DBPROTOCOL(DRDACBF)
/*
```

Example 9-2 shows how to BIND a package with the new option DBPROTOCOL(DRDACBF) on the requester.

Example 9-2 Option DBPROTOCOL(DRDACBF) at BIND

```
//DRDACBF2 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//STEPLIB  DD DSN=DB1AT.SDSNLOAD,DISP=SHR
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
 DSN S(DB1A)
 BIND PACKAGE (N244C) MEMBER(N244C) ACT(REP) ISO(CS) -
 QUALIFIER(USRT001) RELEASE(COMMIT) CURRENTDATA(NO) VALIDATE(BIND) -
   DBPROTOCOL(DRDACBF)
 BIND PACKAGE (DSNDB1B.N244C) MEMBER(N244C) ACT(REP) ISO(CS) -
 QUALIFIER(USRT001) RELEASE(COMMIT) CURRENTDATA(NO) VALIDATE(BIND)
/*
```

9.3.1 Performance measurements

We used a workload that fetches (read only) 500,000 rows from a remote DB2 for z/OS server to a DB2 for z/OS requester to evaluate the benefit of package-based continuous block fetch. The size of a row is approximate 32 KB. The results are listed in Table 9-2.

Table 9-2 Performance results with DBPROTOCOL(DRDACBF)

Time in seconds		DB2 10	DB2 11	Delta %
Server	**CL1 elapsed**	235.492190	47.260880	79.93
	CL2 elapsed	47.944811	34.775680	27.47
	CL1 CPU	33.718104	7.294332	78.37
	CL2 CPU	20.374005	5.241756	74.27

Time in seconds		DB2 10	DB2 11	Delta %
Requester	**CL1 elapsed**	235.540920	189.510000	19.54
	CL2 elapsed	233.821050	188.065100	19.57
	CL1 CPU	75.049996	57.477127	23.41
	CL21 CPU	73.220080	55.932603	23.61
	Messages sent	1,000,000	63	
	Messages received	1,000,000	666,714	
	Bytes sent	120,002,000	8,571	
	Bytes received	298,461.500	297,130,500	

Results show that the new DBPROTOCOL(DRDACBF) helps:

- On the server, both class 1 and class 2 CPU time reduction is about 75%.
- On the requester, both class 1 and class 2 CPU time reduction is about 23%, and elapsed time reduction is about 20%.
- The numbers of messages to be sent from the requester and to be received by the requester reduces significantly. This helps to reduce network traffic and improves response time and throughput.

For DB2 for z/OS to DB2 for z/OS applications that have fetch only query with a large result set, it is suggested to rebind the application packages with new protocol DRDACBF to take advantage of package-based continuous block fetch to reduce CPU utilization and elapsed time.

To maximize the benefit of this enhancement, it is suggested to have at least the requester on z/OS 2.1 or above with PTF UK97808 applied, and to have the SYS1.TCPPARMS TCPCONFIG parameters TCPMAXRCVBUFRSIZE and TCPMAXSENDBUFRSIZE set to 2M.

9.4 Implicit commit for stored procedures

With this enhancement, when autocommit is enabled, DB2 implicitly executes a commit after the stored procedure execution has completed.

If autocommit is enabled at the client (which is the default for both CLI/ODBC and JDBC clients), a commit is implicitly executed at the DB2 for z/OS server once the stored procedure execution has completed and all result set data is prefetched and all result set cursors have been early closed.

This differs from the previous explicit-only support in the following ways:

- No network flow is needed for the sending of a commit message to fulfill an autocommit.
- Threads now become eligible for subsystem reassignment when sysplex workload balancing is enabled.

Note: Previous to the implicit commit for call support, when using the stored procedure parameter 'COMMIT ON RETURN' and assigning it to 'YES', the DBAT would not be eligible for subsystem reassignment at commit when sysplex workload balancing is enabled because the client is not seen by the driver. With the new implicit commit for call support, the DBAT is eligible for subsystem reassignment.

Implicit commit for stored procedure call statements requires DB2 11 NFM and a JCC or CLI client at version 10.5 or later.

9.4.1 Performance measurements

This section documents the results of the performance measurements executed to demonstrate the CPU and elapsed time reduction obtained with implicit commit for stored procedures.

During the tests, the environment was configured as such:

- System zEC12 with three dedicated processors
- z/OS 2.1
- JDBC Type 4 clients over 1 Gb Ethernet network

Single row result set and one output parameter

In this scenario, we call a stored procedure that returns a single result set consisting of one row. Each row consists of a single CHAR(1) column. There is also a single CHAR(1) output parameter. The results are shown in Table 9-3.

Table 9-3 Single row result set and one output parameter performance

Time in microseconds	Class 1 elapsed time	Class 1 CPU time	Class 2 CPU time
Explicit COMMIT	414	136	105
Implicit COMMIT	125	124	106
Reduction %	69.8	9.7	0

This could be considered as a near “best case” scenario for CPU improvement from the new implicit commit support for stored procedures.

Single result set consisting of 10 rows and 1 output parameter

This scenario calls a stored procedure that returns a single result set consisting of 10 rows, each row consisting of a single CHAR(1) column, with a single CHAR(1) output parameter. The results are shown in Table 9-4.

Table 9-4 Single result set consisting of 10 rows and 1 output parameter performance

Time in microseconds	Class 1 elapsed time	Class 1 CPU time	Class 2 CPU time
Explicit COMMIT	633	156	125
Implicit COMMIT	145	143	124
Reduction %	77.1	8.3	0

The 10-row result set increases the total amount of CL1 time, decreasing the percentage improvement that is seen by the implicit commit for one row.

Multiple result sets (2) consisting of 32 rows and five columns each, four input parameters, and one output parameter

This scenario calls a stored procedure that returns two result sets consisting of 32 rows each. Each row has five columns and totals 40 bytes. There are also four input parameters and one output parameter. The results are shown in Table 9-5.

Table 9-5 Multiple result sets performance

Time in microseconds	Class 1 elapsed time	Class 1 CPU time	Class 2 CPU time
Explicit COMMIT	885	248	213
Implicit COMMIT	237	234	214
Reduction %	77.1	5.6	0

IRWW stored procedure workload

The IRWW workload is an OLTP-based workload consisting of seven SQL stored procedures. On average, there are approximately 15 SQL statements per stored procedure execution. The results are shown in Table 9-6.

Table 9-6 IRWW stored procedure workload performance

Time in microseconds	Class 1 elapsed time	Class 1 CPU time	Class 2 CPU time
Explicit COMMIT	13517	782	736
Implicit COMMIT	11976	750	737
Reduction %	11.41	4.1	0

Because of the longer elapsed time to execute a more complex stored procedure, the observed elapsed time savings of one less network trip becomes less noticeable.

9.5 Array data type performance

DB2 11 NFM allows the use of an array data type for SQL scalar functions and procedures. An array type is a user-defined data type that is an ordinary array or an associative array. The elements of an array type are based on one of the existing built-in types.

The CREATE TYPE (array) statement defines an array data type. The SYSIBM.SYSDATATYPES table contains one row for each array data type created.

The array data type can be used only as a data type of:

- An SQL variable.
- A parameter of a native SQL procedure.
- A parameter or RETURNS data type of an SQL scalar function.
- The target data type for a CAST specification.

> **Tip:** Refer to IBM *DB2 11 for z/OS Technical Overview*, SG24-8180 for more details about Array data types and related functions in DB2 11.

9.5.1 Performance measurements

The following performance measurements were conducted to demonstrate potential CPU and elapsed time reduction from using array data types. A medium complexity OLTP stored procedure workload was used for the tests.

The test environment was configured with the following components:

- zEC12 processor with 16 CPs
- DB2 11
- z/OS 1.13
- 120 GB memory
- DB2 Connect 10.5 FP3
- 150 CLI client threads and HiperSockets for network communication

Before DB2 11, input and output parameters consisting of a list of character delimited strings (that is, “AAAA, BBBB, CCCC, DDDD, and so on”) had to be broken up into their individual elements using an external table function. With the new array data type support, the individual character string elements can be passed in and out using an array instead. This saves the CPU and response time overhead of calling an external table function.

This external table function (written in C) was used in 12% of the procedures executed based on a predefined workload mix. So, from an overall standpoint, the benefit will not be as noticeable as it would be on an individual procedure basis.

The overall workload results are shown in Table 9-7.

The overall workload result shows a 5% reduction in DB2 CPU time, and a 6% increase in ITR (normalized CPU/tran). ITR stands for internal throughput rate. It is the external throughput rate (ETR) normalized to 100% CPU. ETR is the observed transactions per second at the measured CPU %.

MSTR, DBM1, and IRLM rows show the CPU/commit for the respective address spaces.

Table 9-7 Overall workload results: Array use

	Non-array	Array	Reduction %
ETR	2910.278	2935.278	-0.86
ITR	6160.622	6528.643	-5.97
System CPU	47.24%	44.96%	4.83
CL1 elapsed time	52.679	52.011	1.27
CL1 CPU time	2.213	2.1	5.11
CL2 elapsed time	52.679	50.562	1.28
CL2 CPU time	2.150	2.045	4.88
MSTR	0.006	0.006	0
DBM1	0.087	0.081	6.90

	Non-array	Array	Reduction %
IRLM	0.001	0.001	0
Total CPU	2.244	2.133	4.95

Up to 40% CPU reduction and 43% elapsed time reduction were observed for some procedures within this workload that previously used an external table function to parse the delimited character strings passed as input/output parameters.

Up to 24% CPU reduction and 7% elapsed time reduction were observed for some procedures that benefitted from array data type support by using new built-in functions, primarily ARRAY_AGG() and UNNEST().

The ARRAY_AGG function returns an array in which each value of the input set is assigned to an element of the array. ARRAY_AGG can be invoked in:

- Select list of a SELECT INTO statement
- Select list of a fullselect in the definition of a cursor that is not scrollable
- Select list of a scalar fullselect as a source data item for a SET assignment-statement (or SQL PL assignment-statement)
- A RETURN statement in an SQL scalar function

Example 9-3 shows a sample CREATE array data type followed by a sample UDF where the array data type is used in a RETURN statement of an SQL scalar function. This sample UDF uses the ARRAY_AGG (aggregate) function and returns an array data type to the caller.

Example 9-3 Array data type create statement and sample use case in a scalar function

```
CREATE TYPE PHONELIST AS CHAR(4) ARRAY[];

CREATE FUNCTION PHONELIST_UDF (LOWSAL DECIMAL(9,2))
      RETURNS PHONELIST
      LANGUAGE SQL
      CONTAINS SQL
      NO EXTERNAL ACTION
RETURN
(SELECT ARRAY_AGG(PHONENO ORDER BY SALARY)
     FROM DSN81110.EMP WHERE SALARY > LOWSAL)
```

The UNNEST table function treats an array like a table to fetch data (that is, rows) from the array. You can use the UNNEST construct (collection-derived table), which returns a result table that contains a row for each element of an array. For instance, using UNNEST, you can retrieve a list of the phone numbers from the array returned by the PHONELIST_UDF, as shown in Example 9-4.

Example 9-4 Sample invocation of UNNEST table function

```
SELECT * FROM dsn81110.emp WHERE phoneno = ANY (SELECT T.PHONE FROM
unnest(phonelist_udf(30000)) AS T(PHONE))
```

9.6 Support for multi-threaded Java stored procedure environment

Many users rely on a DB2 Java stored procedure environment for mainframe modernization, and to run Java applications on z/OS.

Before DB2 11, the Java stored procedure environment would execute 1 JVM per task control block (TCB) in the Workload Manager (WLM)-managed stored procedure environment. This leads to a big storage footprint per TCB, which can have scalability and performance implications.

In DB2 11, the Java stored procedure environment is changed to execute in a multi-threaded model to provide enhanced throughput, a significantly reduced storage footprint, and reduced CPU usage.

9.6.1 Multi-threaded environment

A multi-threaded environment requires a 64-bit JDK at version 1.6 or higher. The JDBC 4.0 standard is also required. Specify JDBCSTD=4 in your environment file.

When running SQLJ stored procedures, the property "db2.jcc.sqljStmtCacheSize=50" is used to cache statements. This is a new property for the multi-threaded environment only, and should be set to a value matching the number of individual statements within the stored procedures running in a particular WLM environment.

A much larger maximum heap can be defined with the new multi-threaded environment if needed, as all TCBs share the same JVM, which is now 64-bit.

The NUMBER OF TCBS (NUMTCB) field determines the number of SQL CALL statements and invocations of user-defined functions that can be processed concurrently in the address space of the default WLM environment.

In the WLM address space startup procedure for Java routines, the MINSPAS value is the minimum number of stored procedure address spaces that WLM starts and maintains. Valid values are 0 - 50. If you specify 0, WLM starts and shuts down stored procedure address spaces as applications require them.

A larger value for NUMTCB can now be defined on the WLM procedure similar to COBOL or C stored procedures. The value for MINSPAS can most likely also be set to 1 (if currently defined).

9.6.2 Performance measurements

JDBC measurements were conducted to demonstrate the CPU, throughput, and storage reduction improvements of the new multi-threaded Java stored procedure environment. The following configuration was used:

- zEC12 processor with 3 CPs
- z/OS 2.1
- JDK 6.0.1 SR4 64-bit and 31-bit
- JCC driver 4.17.16, 15
- JDBC Type 4 clients
- DB2 10 and DB2 11

The following measurements show the performance results when using the new 64-bit multi-threaded environment supported by DB2 11 (which still supports the 31-bit environment) in comparison to the previous 31-bit environment used in DB2 10. The IRWW OLTP workload consisting of seven stored procedures executing approximately 15 SQL statements on average was used for the measurements.

Considerations about NUMTCB:

- For the 64-bit WLM environment, NUMTCB was set to 15. This was to ensure that each JDBC client thread had its own TCB. The NUMTCB value can be set much higher without hitting issues with storage.
- For the 31-bit environment, NUMTCB was set to 5. When using a maximum heap of 128 MB, 5 TCBs are considered safe, but it might also mean that a second WLM environment may be created, or else there may be some queuing on TCBs.

A minimum/maximum heap of 64 MB/128 MB was used for both JDBC and SQLJ. Due to the low complexity of the stored procedures, setting a higher maximum heap in the multi-threaded environment provided no performance benefit.

The results are shown in Table 9-8.

Table 9-8 Multi-threaded Java stored procedure

	ETR (trans per sec)	z/OS CPU %	ITR (tr/sec normalized)	CL1 CPU	CL2 CPU
SQLJ SP 31-bit	1607.42	45.12	3562.54	622	322
SQLJ SP 64-bit multi-threaded JVM	1631.75	43.70	3733.98	620	317
Delta in %			+4.81	-0.32	-1.55
JDBC SP 31-bit	1548.03	57.29	2702.09	864	429
JDBC SP 64-bit multi-threaded JVM	1606.53	52.74	3046.13	823	416
Delta %			+12.73	-4.98	-3.13

In summary:

- With JDBC, class 1 DB2 CPU per transaction is reduced by 4.98%, while normalized throughput increases by 12.73%.
- With SQLJ, DB2 class 1 CPU per transaction is reduced by 0.32% and normalized throughput is increased by 4.81%.
- WLM address space real storage usage is decreased from 680 MB to 320 MB while using multi-threaded JVM.

9.7 Optimizations for calling stored procedures with local JDBC/ODBC drivers

DB2 11 delivers performance optimizations when calling stored procedures from a local ODBC/JDBC driver. The enhancements do not require changes to the application and are available in DB2 11 CM.

The following list provides the enhancements:

- Result sets returned from the stored procedure are returned using limited block fetch (LBF) and progressive streaming.
- Result sets now implicitly closed after result set is exhausted (SQLCODE +100).
- CALL and DESCRIBE PROCEDURE statements are bundled into a single message reducing trips between the driver and DBM1.
- ALLOCATE CURSOR and DESCRIBE CURSOR statements are also bundled.

These optimizations are specific to the local JDBC/ODBC drivers. For both ODBC and JDBC drivers in DB2 11, limited block fetch is enabled by default.

9.7.1 JDBC performance measurements

JDBC performance measurements were conducted to demonstrate the CPU and elapsed time reduction of the new ODBC/JDBC driver optimizations. The test environment configuration consisted of the following components:

- z196 processor with 3 CPs
- z/OS 1.13
- Local JDBC/ODBC client using RRS attach
- DB2 11

Limited block fetch guarantees the transfer of a minimum amount of data in response to each request from the requesting system. With limited block fetch, a single conversation is used to transfer messages and data between the requester and server for multiple cursors.

JDBC driver performance results

The following scenarios were measured:

- One result set, one row, CHAR(1) column
- One result set, 3000 rows, CHAR(1) column
- Two result sets, 3000 rows, CHAR(1) column
- One result set, 3000 rows, VARCHAR(32000) column, 200 bytes of data
- One result set, 3000 rows, CLOB(2M) column, CLOB assigned 2 MB of data
- One result set, 50 rows, CLOB(2M) column, 2 MB of data
- Three result sets, one row, CHAR(1) column: to show benefit of bundling

Figure 9-3 on page 224 summarizes the performance results when enabling limited block fetch using varying column types, sizes, and numbers of result sets.

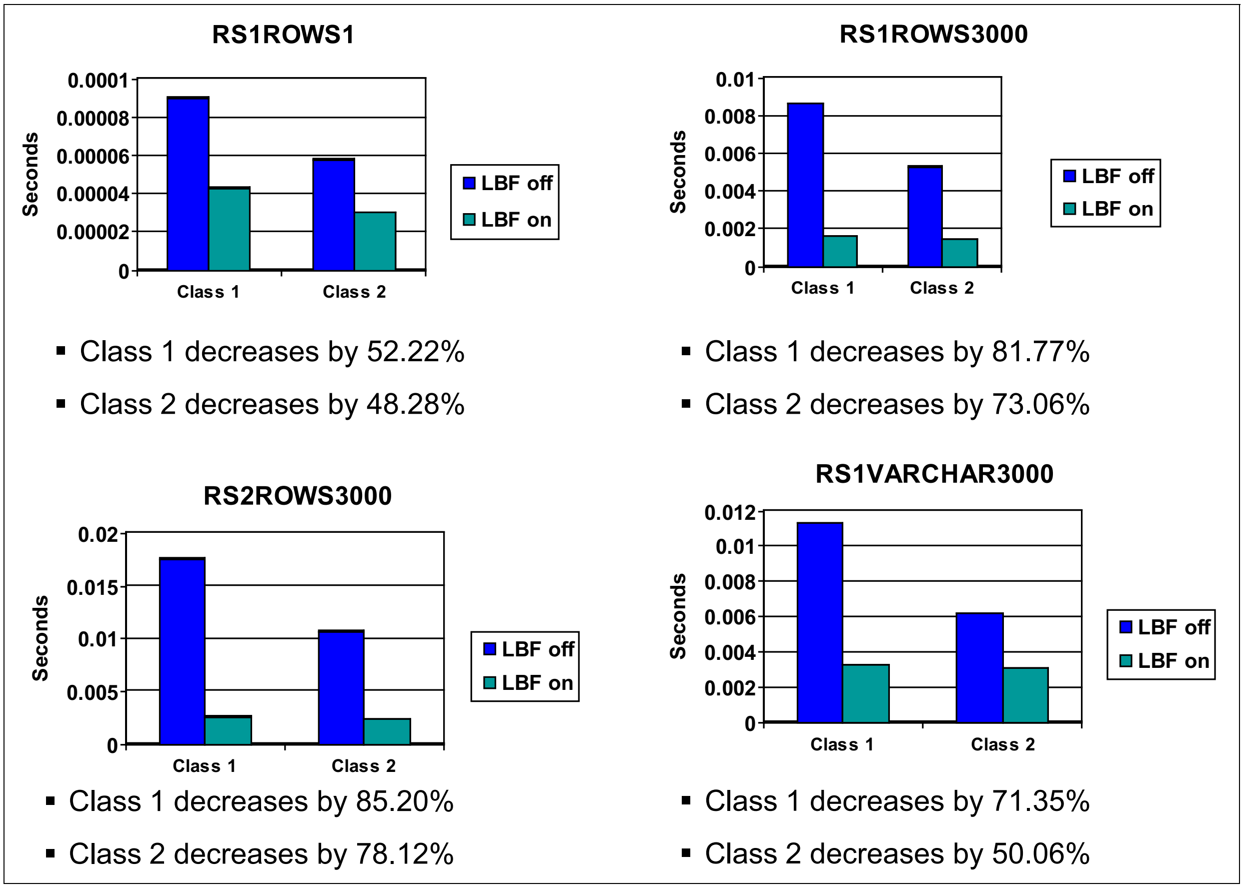

Figure 9-3 Enabling limited block fetch using varying column types, sizes, and numbers of result sets: JDBC

ODBC driver performance results

The following scenarios were measured:

- One result set, one row, CHAR(1) column
- One result set, 3000 rows, CHAR(1) column
- Two result sets, 3000 rows, CHAR(1) column
- One result set, 3000 rows, VARCHAR(32000) column, 200 bytes of data
- One result set, one row, BLOB(1 K) column, BLOB assigned 1 K of data
- One result set, 50000 rows, BLOB(1 K) column, BLOB assigned 1 K of data
- Three result sets, one row, CHAR(1) column: to show benefit of bundling

Figure 9-4 on page 225 shows the ODBC performance results when enabling limited block fetch using varying column types, sizes, and numbers of result sets.

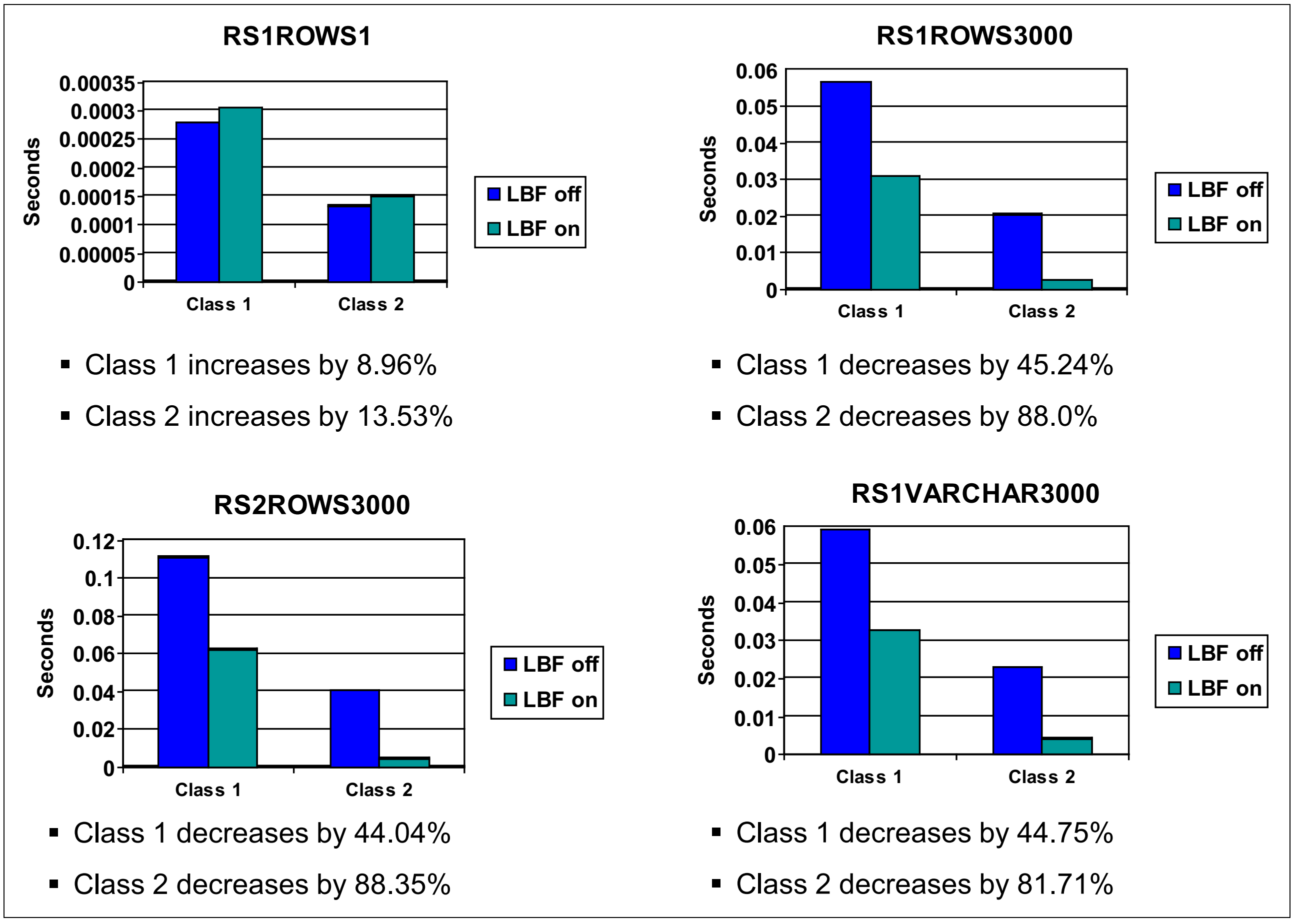

Figure 9-4 Enabling limited block fetch using varying column types, sizes, and numbers of result sets: ODBC

9.8 Distributed IRWW workload measurements

For the distributed access to DB2 for z/OS, we used the distributed IRWW warehouse, workloads. This is an OLTP workload that consists of seven transactions. Each transaction consists of one to many SQL statements, each performing a distinct business function in a predefined mix. See “IRWW distributed workload” on page 299.

The evaluation of the distributed access to DB2 for z/OS focuses on the differences between DB2 10 and DB2 11. Class 1 elapsed time and the total transaction CPU time per commit are the main metrics. The tests were evaluated in the following test environment:

- zEC12 z/OS LPAR: 3 CPs, 32 GB, z/OS 2.01.
- z Linux on System z LPAR: 2 CP with DB2 Connect and DB2 for Linux, UNIX, and Windows 10.5 (s130428).
- JCC driver 3.66.33, JDK 1.7.0, IBM J9 VM (build 2.6, JRE 1.7.0 Linux s390x-64 20121024_126071).
- IBM HiperSockets communication between the two LPARs.

During the test, we executed and reported the performance of the following workloads:

- SQCL: CLI (dynamic) using DB2 Connect.
- JDBC: JDBC (dynamic) using JCC T4 driver.
- SQLJ: Static Java using JCC T4 driver.
- SPCB: Stored procedures in COBOL (static) using DB2 Connect.
- SPSJ: Stored procedures in SQLJ (static) using JCC T4 driver.
- SPNS: Stored procedures in native SQL (static) using JCC T4 driver.

All packages were bound using the RELEASE(DEALLOCATE) BIND option, and we controlled the high performance DBAT by using the `MODIFY DDF PKGREL` command.

The PKGREL(BINDOPT) option was used to use the RELEASE(DEALLOCATE) high performance DBAT capability. The PKGREL(COMMIT) option was used to use the RELEASE(COMMIT) behavior.

9.8.1 Performance measurements

The reported total transaction CPU for SQCL, JDBC, SQLJ, and SPNS workloads is the sum of System Services Address Space, Database Services Address Space, IRLM, and DDF Address Space CPU time per commit as reported in IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS (OMEGAMON PE) statistics report Address Space CPU section. The reported total transaction CPU for stored procedure workloads such as SPCB and SPSJ is the statistics total address space CPU per commit plus accounting class 1 STORED PRC CPU in the OMEGAMON PE accounting report.

Non-stored procedure workloads

Table 9-9 presents delta in % of CL1 elapsed time, DDF per commit, and total CPU time per commit for three non-stored procedure workloads of the distributed IRWW workload suite. The results show up to 16% improvement in total CPU, and up to 11% reduction on CL1 elapsed time. This includes the benefit of DDF performance Synchronous Receive support that DB2 11 offers compared to DB2 10.

Table 9-9 DB2 10 versus 11 compared for distributed IRWW, non-stored procedure workload

Delta %	PKREL(COMMIT)			PKREL(BNDOPT)		
Workload	**CL1 elapsed time**	**DDF/COMMIT**	**Total CPU/COMMIT**	**CL1 elapsed time**	**DDF/COMMIT**	**Total CPU/COMMIT**
SQCL	4.27	9.66	9.54	6.12	11.28	11.22
JDBC	9.91	14.62	14.91	11.02	15.95	16.22
SQLJ	0.56	7.37	7.32	1.34	8.37	8.26

Stored procedure workloads

Table 9-10 on page 227 presents delta of CL1 elapsed time, DDF per commit, and total CPU time per commit of three stored procedure workloads of the distributed IRWW workload suite. The implementation of these stored procedures does not use *implicit commit*, which is a new feature in DB2 11. The results show that there is a small improvement that varies 2 - 4% in reduction of total CPU, and up to 3% reduction of CL1 elapsed time in DB2 11 compared to DB2 10.

Table 9-10 DB2 10 versus DB2 11 compared for distributed IRWW, stored procedure workload without implicit commit

Delta %	PKREL(COMMIT)			PKREL(BNDOPT)		
Workload	**CL1 elapsed time**	**DDF/COMMIT**	**Total CPU/COMMIT**	**CL1 elapsed time**	**DDF/COMMIT**	**Total CPU/COMMIT**
SPNS	0.79	1.80	2.24	3.26	1.32	3.03
SPCB	3.20	1.84	4.03	3.26	1.76	4.04
SPSJ	1.74	0.57	2.51	1.58	0.25	2.45

Stored procedure workloads with implicit COMMIT

Table 9-11 presents delta of CL1 elapsed time, DDF per commit, and total CPU time per commit for three stored procedure workloads of the distributed IRWW workload suite. The implementation of these stored procedures includes implicit commit for stored procedures, which is a new feature in DB2 11. The results show that implicit commit for stored procedure improves performance by reducing cost in the DB2 engine and the DDF layer. With stored procedure implicit commit support, CL1 elapsed time is reduced 5 - 8%, and total stored procedure CPU time per commit is reduced up to 8% in DB2 11 compared to DB2 10.

Table 9-11 DB2 10 versus DB2 11 compared for distributed IRWW, stored procedure workload with implicit commit

Delta %	PKREL(COMMIT)			PKREL(BNDOPT)		
Workload	**CL1 CPU time**	**DDF/COMMIT**	**Total CPU/COMMIT**	**CL1 CPU time**	**DDF/COMMIT**	**Total CPU/COMMIT**
SPNS	5.70	6.28	5.28	7.05	7.77	6.29
SPCB	8.47	12.88	8.64	8.02	12.58	7.45
SPSJ	5.76	12.50	6.09	5.31	12.27	5.81

9.8.2 Conclusion

DB2 11 offers CPU time and elapsed time reductions over DB2 10. Comparing the same non-stored procedure distributed workloads, there is up to 16% reduction in total CPU usage, and up to 11% reduction on CL1 elapsed time. For stored procedure workloads that do not use the implicit commit feature in DB2 11, the total SP CPU time per commit reduction is up to 4%, and up to 3% reduction on CL1 elapsed time. For stored procedure workloads that use the new implicit commit feature in DB2 11, the total SP CPU time per commit reduction is up to 8%, and the CL1 elapsed time reduction is up to 8%.

10

Utilities

DB2 11 includes various improvements to utilities. The utilities are enhanced to support all new functions in DB2 11, including the pervasive extended RBA/LRSN. The support includes more widespread use of the System z platform functions, such as more System z Integrated Information Processor (zIIP) exploitation (see 3.1.2, “Additional zIIP redirect for the DB2 utilities” on page 67) and I/O. DB2 11 utilities also show the trend to simplifying data management, resource consumption, and maximize availability.

In this chapter, we describe the results of performance measurements for the following functions:

- Load parallelism
- REORG switch phase
- Partitioned Online REORG with NPSIs
- Statistics collection
- Partition-level image copies
- Cross Loader
- Utility MRU
- Suppress NULL index support

10.1 Load parallelism

The DB2 LOAD utility has long supported parallelism when the input data is organized in multiple data sets. Using multiple data sets enables DB2 to overcome any I/O bottlenecks associated with reading the input data sets. DB2 11 supports parallelism even when all of the data is in a single input data set. Dividing the input records into different data sets sorted by partition is still the preferred method, but if the data is originally stored as a single data set, it is no longer necessary to segregate the data by partition. Input I/O bottlenecks are not overcome, but there are many cases where input I/O is no longer a bottleneck.

Figure 10-1 shows the syntax diagram for the PARALLEL option. To use parallelism with a single input data set, the PARALLEL keyword must be specified on the LOAD statement. The degree of parallelism can also be specified as the number of parallel subtasks.

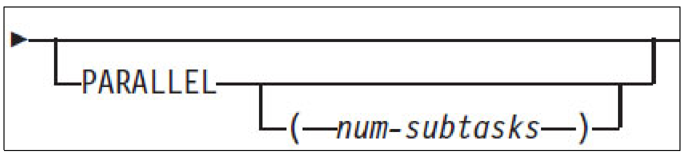

Figure 10-1 PARALLEL option

For a single input data set, the PARALLEL option specifies the maximum number of subtasks that are to be used in parallel when the utility loading a table space from a single input data set and building the indexes. By using parallel subtasks, the utility can potentially reduce the elapsed time of the load operation.

For multiple input data sets, where there is one data set for each partition, the PARALLEL option specifies the maximum number of subtasks that are to be used with loading the data partitions, building the indexes, and gathering statistics. If 0 or no value is specified for num-subtasks, the LOAD utility calculates the optimal number of parallel subtasks.

The LOAD PARALLEL option enables parallelism for the following table space types: simple, segmented, classic partitioned, partition by range. Parallelism is not supported for partition by growth table spaces.

The default degree of parallelism with SHRLEVEL NONE is equal to the number of CPs (not zIIPs) on the LPAR. The degree with SHRLEVEL CHANGE is equal to three times the number of CPs. Typically, one should expect an increase of 0 - 10% CPU time due to the overhead of scheduling the work to parallel tasks while elapsed time gets reduced.

This new type of parallelism can generally be thought of as CPU parallelism because so much of the benefit comes from the exploitation of multiple CPs. However, some of the benefit comes from parallel index synchronous I/Os.

Some types of LOAD processing are CPU bound. For example, delimited input requires considerable CPU overhead for processing the input. CCSID conversion is also expensive. There is also much CPU overhead to convert “external” numerical representations of data (of any CCSID) into a DB2 format. Even VARCHAR requires some extra CPU overhead to validate the length constraints. Another case is the case of many columns in a table. Unlike REORG, which processes at a record level, Load processes at a column level. So having tens of columns or hundreds of columns does not affect REORG CPU time but Load CPU time will be significantly affected.

In general, the LOAD utility does not benefit from CPU parallelism if row compression is not used, if no numeric conversions are involved, if no complex data types are involved (such as

DATE/TIME columns), if the input is not delimited, and if all text columns are fixed length CHAR. The more indexes there are, the more benefit there is.

The LOAD utility benefits from index synchronous I/O parallelism if there were a lot of synchronous I/Os to the cluster index. However, *index I/O parallelism* that was introduced in DB2 10 already enables INSERT index I/O parallelism when there are three or more indexes. can also reduce the elapsed times of LOAD RESUME YES SHRLEVEL CHANGE utility executions because the utility functions similar to a MASS INSERT when inserting to indexes.

Since LOAD SHRLEVEL CHANGE generally has more CPU overhead than LOAD SHRLEVEL NONE, the value of CPU parallelism tends to be greater for SHRLEVEL CHANGE than it is for SHRLEVEL NONE. Following is an explanation of why SHRLEVEL CHANGE has more CPU overhead than SHRLEVEL NONE.

Using SHRLEVEL NONE, the new rows are not stored in cluster sequence. The utility constructs a page full of rows and then feeds the constructed page to data manager. Data manager then appends the new page to the end of the table and updates the indexes accordingly. Since the pages are built one at a time, there is only one data getpage for each page of data. The utility calls data manager only once per data page and data manager need not call index manager to use the cluster index to determine a candidate page.

In contrast, using SHRLEVEL CHANGE works the same way that SQL inserts work. That is, the utility calls data manager once per row and for each row data manager must call index manager to determine the candidate page to store the row. If the keys in the input records are presorted, the inserts are sequential or at least skip sequential, and there is a single getpage for each modified data page. Sorting the input records may also reduce the number of index getpages. Consequently, whether or not the input records are sorted affects the value of CPU parallelism.

If you specify the PARALLEL keyword and SHRLEVEL CHANGE, set the LOCKSIZE attribute of the table space to ROW to minimize contention on the parallel subtasks.

The effect of parallelism with LOAD REPLACE is the same as LOAD SHRLEVEL NONE when the table space is initially empty.

10.1.1 LOAD SHRLEVEL NONE

In this section, we provide a summary of the measurements of parallelism using LOAD SHRLEVEL NONE compared to no parallelism. The parallelism is determined by DB2 based on the number of CPs and storage constraints.

The test case loaded 100 million rows from a single input file starting from an empty table. In each of the following measurement test cases, the table space was defined with 20 partitions, without compression. The measurements were done using a z196 processor, 1 zIIP, 100,000 data buffers, and no indexes. The rows consisted of 14 fixed length CHAR columns, five decimal columns, and six integer columns.

The first set of tests used no indexes and no row compression.

The results are shown in Table 10-1.

Table 10-1 LOAD SHRLEVEL NONE, no indexes, no compression

Run #	Input	Conversion	CPU %	Elapsed time %
1	Sorted	None	+12.5	+6
2	Sorted	Numeric	+4	-48
3	Unsorted	None	+5	-43
4	Unsorted	Unicode	+2	-71

The results verified that parallelism benefits the most when CCSID conversion is done (run 4), or when numeric conversion is done (run 2). Also, when the input data was random regarding the partitions (run 3), there was a significant improvement even when there was no conversion. In each of these cases, there was a small increase in CPU cost with a large decrease in elapsed time. When there was no CCSID conversion or numeric conversion, parallelism hurt both CPU performance and the elapsed time. That is because the parallel tasks cannot insert into the same partition at the same time.

For SHRLEVEL NONE measurements (runs 5 - 8) in Table 10-2, compression was used and 100 million rows were loaded into an empty table. Sometimes an index was used and sometimes six indexes were used. Sometimes the input data set was sorted by partition (but unsorted within a partition), and sometimes the input records were random regarding the cluster key. For the eight test case, the CHAR columns were changed to VARCHAR.

Table 10-2 LOAD SHRLEVEL NONE, with indexes and compression

Run #	Input	Conversion	# Indexes	CPU %	Elapsed time %
5	Sorted	None	0, 1	+10	-10
6	Unsorted	None	1	0	-19
7	Sorted, unsorted	None	6	0	-15
8	Sorted	Numeric, varchar	6	0	-49

The results showed that the most benefit occurred when there was numeric conversion and the text columns were VARCHAR. In the other cases, the improvement was modest.

10.1.2 LOAD SHRLEVEL CHANGE

The next set of measurements used LOAD SHRLEVEL CHANGE, starting with a table that was seeded with 1 million rows. Again the utility added 100 million rows. See Table 10-3.

Table 10-3 LOAD SHRLEVEL CHANGE

Run #	Freespace %	Page level locking	# Indexes	CPU %	Elapsed time %
1	0	No	0	+29	-4.7
2	0	No	6	+25	-65
3	10	Yes	6	+9	-71

The benefit of parallelism is sensitive to the same factors that affect SQL insert parallelism. Row level versus page level locking is one of those factors. Free space is another factor. They can reduce potential contention among the parallel tasks all trying to load into the same partition.

Finally, a single input data set was compared to 20 input data sets using LOAD SHRLEVEL CHANGE with six indexes and no free space, with the table seeded with 1 million rows as before. The input data was not presorted within a partition. Using a single input data set had the same elapsed time as multiple data sets, but used 20% more CPU time. The reason that the elapsed time did not change is that the elapsed time was gated by the time to create the 20 partitions, which perform the same regardless of how the data gets loaded.

See Table 10-4.

Table 10-4 LOAD SHRLEVEL CHANGE, not sorted, no free space

Run #	# input data set	Freespace %	# Indexes	CPU %	Elapsed time %
1	1 versus 20	0	6	+20	0

Comparing multiple input data sets (one per partition) versus a single input data set shows that in general the elapsed time is the same, but the CPU time is higher with single input. LOAD is faster if the data has been presorted into multiple data sets. However, there may be some cost to presorting the data, which must be done by a batch program splitting the data possibly using DFSORT, REXX, and a COBOL program. Being able to use a single input improves DB2 availability.

10.2 REORG switch phase

Completing the SWITCH phase of online REORG with SHRLEVEL REFERENCE or CHANGE has often been cited as being of more importance than either the CPU consumption or the elapsed time of the job because sometimes the user must face the choice of either allowing the REORG job to fail or potentially impact applications to allow the SWITCH phase to complete.

The total application-impact time for standard REORG SHRLEVEL CHANGE consists of the time that is taken to acquire the drain, then the time needed to apply the remaining log records in the last log iteration of the LOG phase, followed by the duration of the SWITCH phase. Therefore, the SWITCH phase duration is a subset of the entire drain duration.

The time that is required to acquire drains on target objects is dependent on the number of objects being reorganized. The time allotted to this process is limited by the DRAIN_WAIT parameter. If this is set too low, the REORG utility may fail due to inability to acquire the drain within the allotted time. If however, this parameter is set too high, application claims can time out due to elongated drain times within REORG.

The time taken to apply the remaining log records in the last log iteration is largely dependent upon the setting of parameter MAXRO. If this value is set too high, the REORG utility may attempt to acquire the drain too early, resulting in an excessive number of log records to be applied when the drain succeeds.

Finally, the duration of the SWITCH phase is not controlled by any given REORG parameter. Instead, it is dependent upon the number of objects being processed by the REORG, which is one reason why it is preferable to REORG a few partitions at a time. Sometimes it is necessary to REORG the entire partition table space at one time. For that reason, it is also desirable to minimize the number of partitions by using very large partitions. With DSSIZE set to 64 G, you can have only 256 partitions anyway, and with DSSIZE set to 256 G, you can have only 64 partitions.

The issues at stake are the desire to acquire a drain on the target objects in a timely fashion and, when the drain has been acquired, the desire to hold the drain for as short a time as possible.

DB2 11 provides relief for this problem with three changes:

- Provide drain relief for REORG on subset of data partitions by changing the current serial draining algorithm, to setting drain in control against all target parts before waiting for existing claimers to commit
- Optimize and remove unnecessary physical opens and closes
- Optimize other getpages and subprocesses including SYSLGRNX usage, catalog and directory updates, and so on.

Measurements have been done of REORG SHRLEVEL CHANGE on four-way zEC12 processors in a non-data sharing system using a universal table space (UTS) with 100 million rows spread across 100 partitions. The results are shown in Figure 10-2. See Figure 10-2.

DB2 11 reduced the duration time of the drain and the switch phase by 83%.

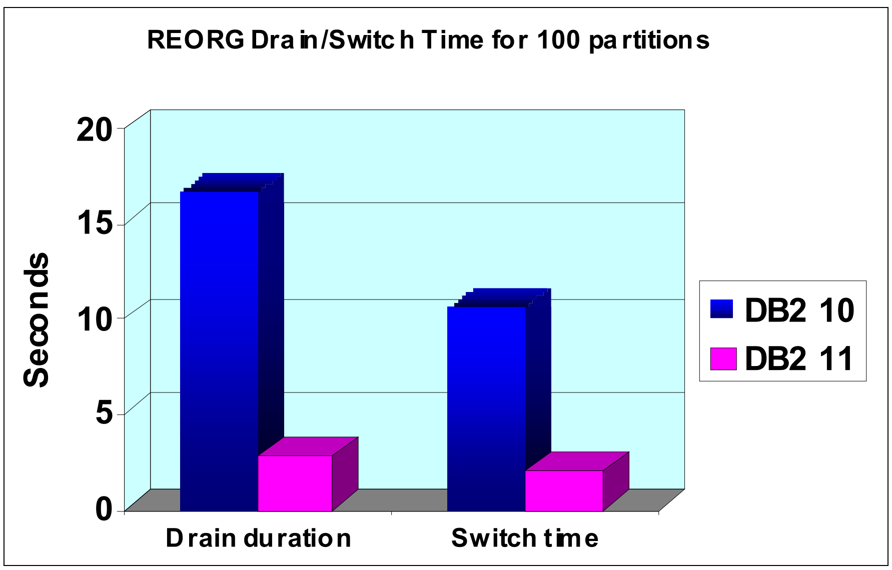

Figure 10-2 REORG Drain/Switch time for 100 partitions

To see the effects of the number of partitions, the number of partitions was varied 20 - 4000 for the same number of rows and the results are displayed in Figure 10-3 on page 235. DB2 11 achieved about the same level of improvement for any number of partitions.

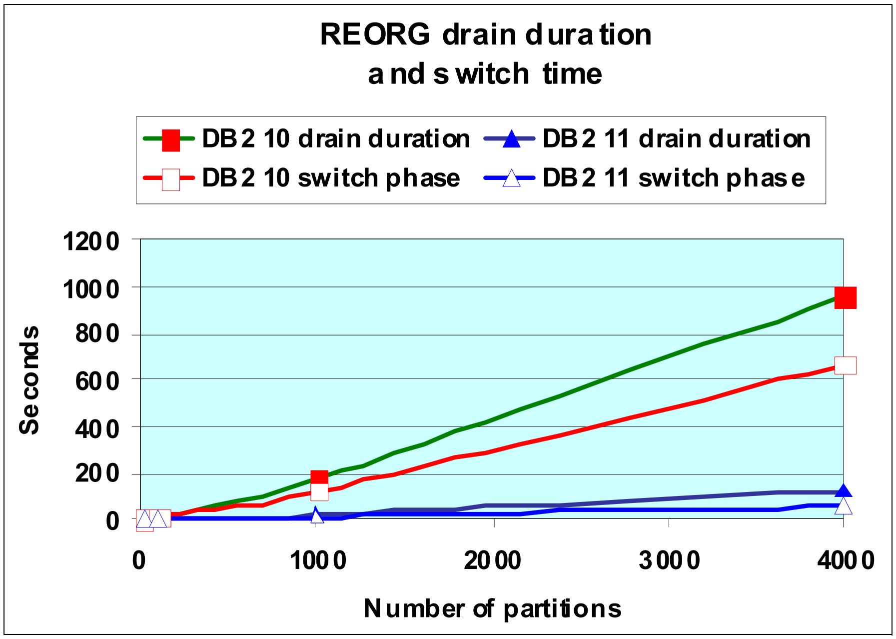

Figure 10-3 REORG Drain/Switch time increasing partitions

10.3 Partitioned Online REORG with NPSIs

Since the removal of the BUILD2 phase for partition-level REORG in DB2 9, the performance of REORG was degraded in some cases due to the cost of building shadow NPSIs (nonpartitioned secondary indexes). Shadow NPSIs are populated initially with keys of parts, which are not in the scope of the REORG during the UNLOAD phase. Then, keys from parts within the scope of the REORG are sorted and inserted into the shadow NPSI during the SORT and REBUILD phases, respectively. Significant performance improvement can be achieved by sorting all keys of the NPSI in the same sort operation and rebuilding the index from the entire set of sorted keys. This enhancement adds functionality to improve performance while leaving the old behavior intact and allows the customer to control the behavior through the installation panel and a keyword. This enhancement modifies the processing of nonpartitioned secondary indexes for REORG TABLESPACE PART SHRLEVEL CHANGE/REFERENCE when NPSIs are defined on the table space. Processing of NPSIs in this case will now be done in either one of the two following ways:

- During UNLOAD, one or more subtasks unload NPSI keys from parts not within the scope of the REORG and build the shadow NPSI. Keys from parts within the scope of the REORG are generated from the reorganized data rows, sorted, and inserted in the shadow index.

  This is the traditional technique.

- During UNLOAD, one or more subtasks may process NPSI keys from parts not within the scope of the REORG. These keys are routed to a sort process to be sorted with the keys from parts within the scope of the REORG. The shadow NPSI is built from this sorted set of keys.

  A new keyword SORTNPSI is added to control the method of building an NPSI during REORG TABLESPACE PART with the following options:

  Option NO Specifies that only keys of the nonpartitioned secondary indexes that are in the scope of the REORG are sorted. This is the traditional technique.

Option YES Specifies that all keys of the nonpartitioned secondary indexes are sorted. This is the new optional technique with a potential significant reduction in elapsed time in some cases.

Option AUTO Is intended to let DB2 choose the generally optimal technique based on available information.

For example, the traditional SORTNPSI NO would be preferred when the percentage of data to be reorganized is small, but the new SORTNPSI YES would be preferred when the percentage of data to be reorganized is large. The traditional SORTNPSI NO is also preferred if the overall system CPU utilization is high, since the new SORTNPSI YES tends to increase the CPU utilization further, possibly leading to a longer wait time for busy CPUs.

A given user might be able to arrive at a better decision based on the specific knowledge of the hardware or software environment.

If the SORTNPSI keyword is not specified, the subsystem parameter REORG_PART_SORT_NPSI determines the value. This subsystem parameter can have AUTO, YES, or NO, with the default setting of AUTO. The benefit of sorting all keys of a nonpartitioned secondary index increases as the ratio of data reorganized to total data in the table space increases. This keyword is ignored for a REORG that is not partition-level or a REORG without nonpartitioned secondary indexes.

This DB2 11 enhancement was retrofitted to DB2 9 and DB2 10 for z/OS via PM55051.

There was another V9/V10 APAR PM87403 in June 2013, which can primarily reduce the CPU time for SORTNPSI NO case (traditional technique) only by reducing the exhaustive index space map page search when allocating new index pages during REORG TABLESPACE SHRLEVEL REFERENCE or CHANGE by PART as well as LOAD RESUME YES with indexes.

REORG of 5% of the data with PM87403

The graph in Figure 10-4 on page 237 shows the zEC12 CPU time in seconds for three cases of partition REORG with four NPSIs, five NPSIs, and six NPSIs when 5% of the total data in the table space is REORG'd. For example, five partitions out of 100 partition table spaces with even-sized partitions. This is also equivalent to one larger partition, which takes up 5% of the total data in 100 partition table space within a partition.

Roughly two times CPU time reduction is observed in this case with PM87403 APAR compared to without PM87403.

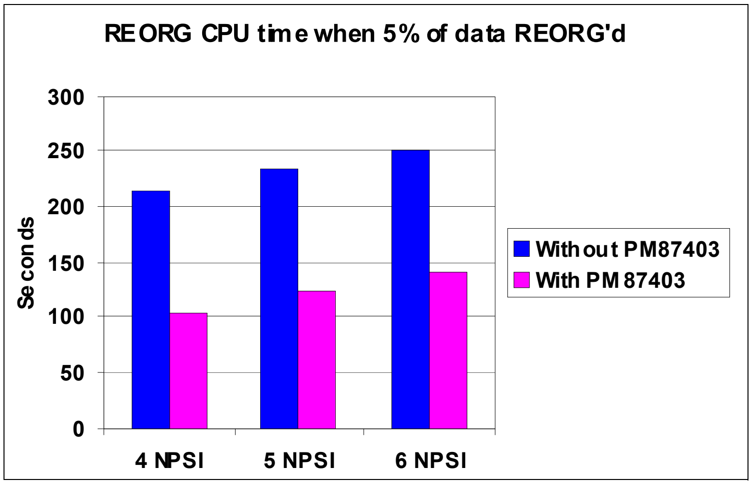

Figure 10-4 REORG CPU time in seconds when 5% of data REORG'd

The corresponding graph for the elapsed time is shown in Figure 10-5. There is an expected reduction in elapsed time corresponding to the elapsed time contribution by the CPU time reduction.

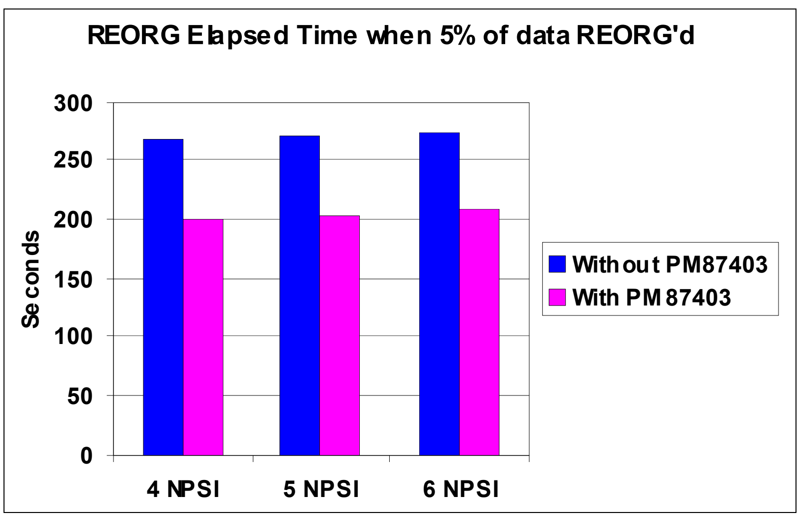

Figure 10-5 REORG elapsed time in seconds when 5% of data REORG'd

REORG of 0.4% of the data with SORTNPSI option and PM87403

Some recent sample measurement comparisons between SORTNPSI NO (traditional technique), with PM87403 applied to improve SORTNPSI NO case, and SORTNPSI YES (new technique) are shown here.

The graph in Figure 10-6 on page 238 shows the zEC12 CPU time comparison and the graph in Figure 10-7 on page 238 shows the elapsed time comparison between SORTNPSI NO (traditional technique) in solid blue and SORTNPSI YES (new option) in dotted red for 0.4% of data REORG'd.

Because of a low percentage of data REORG'd, SORTNPSI YES encounters a significant CPU time increase, especially as the number of NPSI increases.

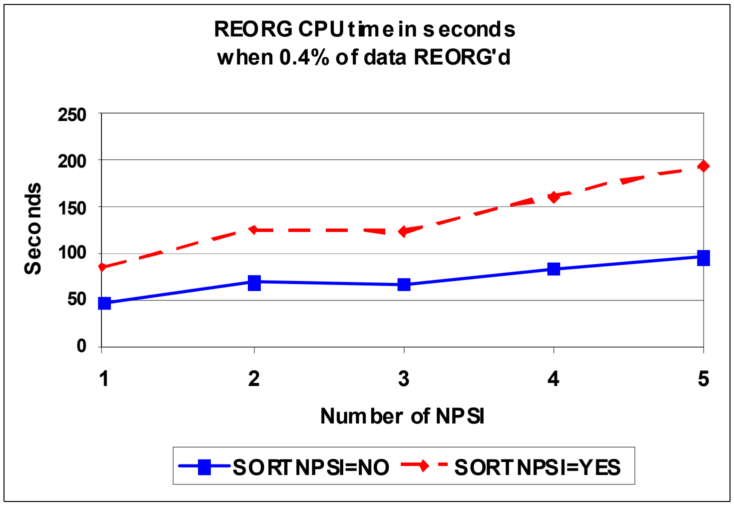

Figure 10-6 REORG CPU time in seconds when 0.4% of data REORG'd

There is not much difference in elapsed time here as shown in Figure 10-7, indicating that for 0.4% of data REORG'd, the elapsed time benefit from the new SORTNPSI YES option is negligible while the CPU time overhead is significantly higher.

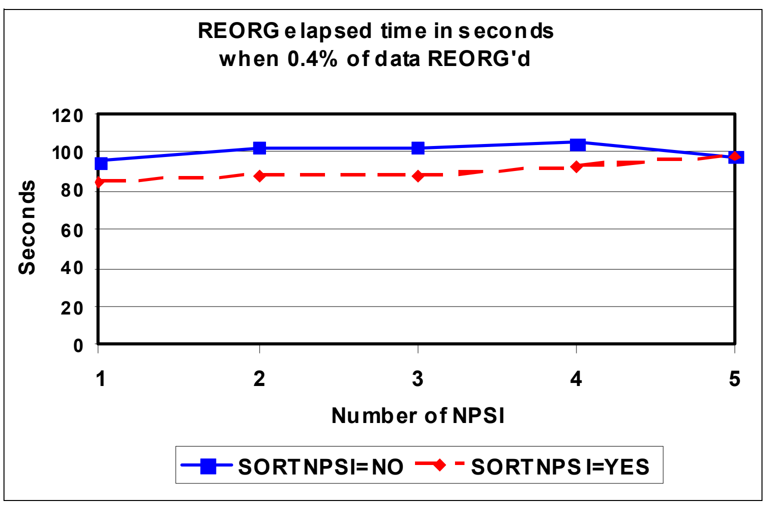

Figure 10-7 REORG elapsed time in seconds when 0.4% of data REORG'd

REORG of 1% of the data with SORTNPSI option and PM87403

The next two graphs (see Figure 10-8 on page 239 and Figure 10-9 on page 239) show the zEC12 CPU time and the elapsed time comparison between SORTNPSI NO (traditional technique) in solid blue and SORTNPSI YES (new option) in dotted red for 1% of data REORG'd.

With a higher percentage of data REORG'd, more benefit of SORTNPSI YES is observed as the elapsed time reduction becomes more significant. Because there is still a significant CPU time increase, the use of SORTNPSI YES needs to be carefully evaluated for each specific case as it can bring a big elapsed time increase rather than reduction in an environment with high CPU utilization from other workloads running.

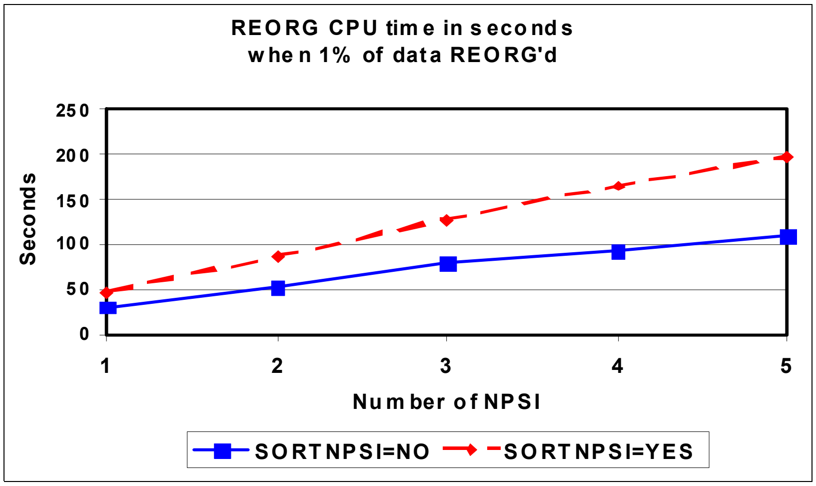

Figure 10-8 REORG CPU time in seconds when 1% of data REORG'd

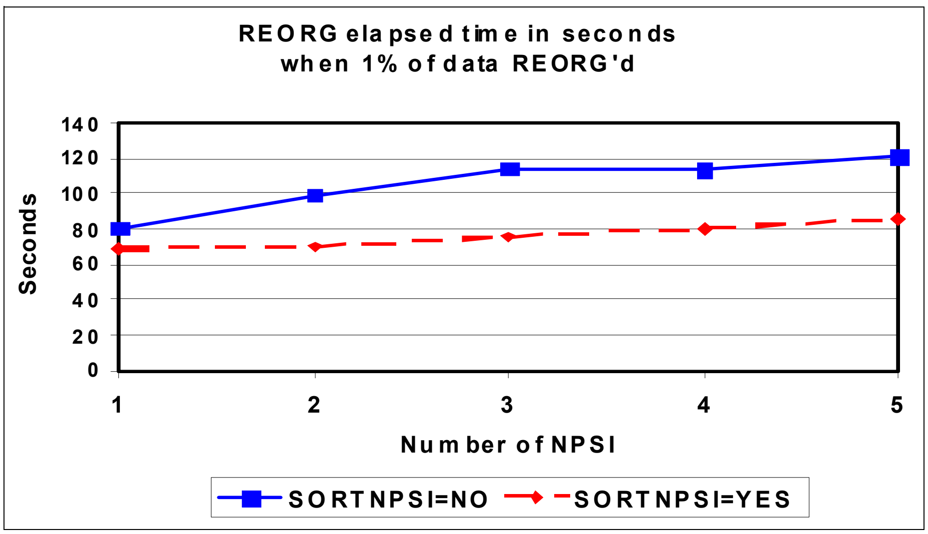

Figure 10-9 REORG elapsed time in seconds when 1% of data REORG'd

REORG of 5% of the data with SORTNPSI option and PM87403

The last two graphs (see Figure 10-10 on page 240 and Figure 10-11 on page 240) show the zEC12 CPU time and the elapsed time comparison between SORTNPSI NO (traditional technique) in solid blue and SORTNPSI YES (new option) in dotted red for 5% of data REORG'd.

With 5% of data REORG'd, the new SORTNPSI YES is faster by two to three times, although the CPU time becomes higher by 0 - 45% for 1 - 5 NPSIs. Here, the new SORTNPSI YES should be preferable possibly except when there are many NPSIs and the overall system CPU utilization is high.

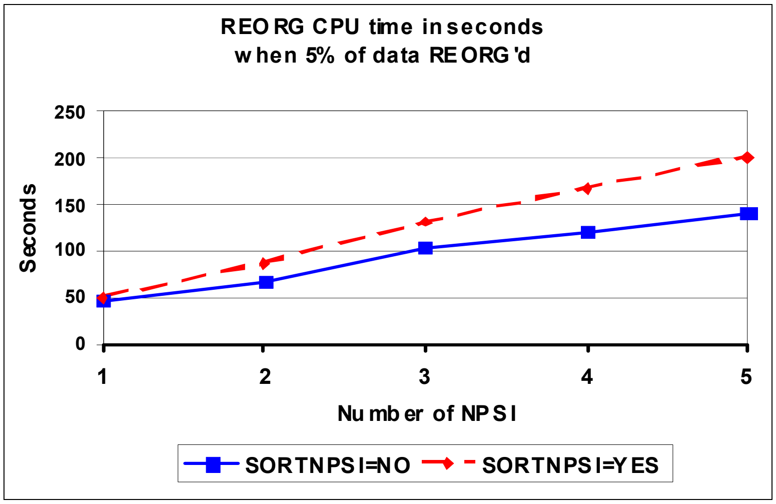

Figure 10-10 REORG CPU time in seconds for 5% of data REORG'd

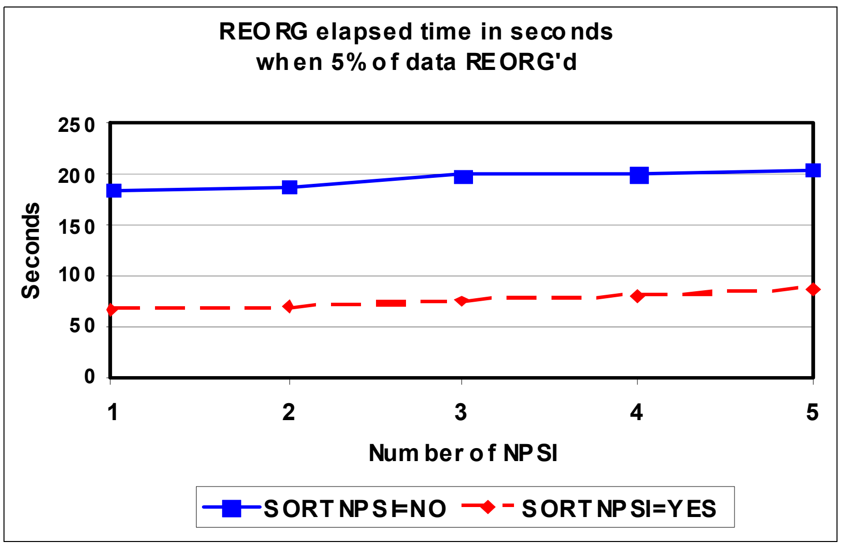

Figure 10-11 REORG elapsed time in seconds when 5% of data REORG'd

Summary

This section explains the potential performance impact of the new SORTNPSI option, with a default of AUTO, to help determine the optimal setting for online REORG of a subset of partitions in a table space, trying to consider environment specifics. The considerations do not apply if an online REORG of the entire table space is executed rather than a subset of partitions.

Based on these measurements and some additional measurements with the percentage of REORG'd data varying from 0.4% to 1%, 2%, 5%, 10%, 15%, 20%, and 25% and with NPSIs varying 1 - 10, the following general rule-of-thumb recommendations can be made on when to use the new technique (SORTNPSI YES) instead of the traditional technique (SORTNPSI NO):

- SORTNPSI NO is generally recommended when less than 0.5% of data is REORG'd.
- SORTNPSI YES is generally recommended when more than 4% of data is REORG'd.
- Between 0.5% and 4% of data to be REORG'd represents a boundary condition.

In this range, a significant increase in CPU time along with a significant reduction in elapsed time is possible by switching to SORTNPSI YES from NO. A decision should be made based on what is more important: CPU time or elapsed time. In an environment with high CPU utilization, a further significant increase in CPU time can lead to a significant increase in elapsed time as well since the wait time for busy CPU can become dominant. In this case, staying at SORTNPSI NO would be recommended.

Another factor that comes into the picture here is the number of NPSIs. In general, SORTNPSI NO is better with more NPSIs. For example, SORTNPSI NO is recommended if the following scenarios exist:

- Roughly 1% of data is REORG'd and NPSIs greater than around 4
- Roughly 2% of data is REORG'd and NPSIs greater than around 10

10.4 Statistics collection

DB2 11 RUNSTATS improves zIIP offload capability of distribution statistics collection function with up to 81% CPU being zIIP-eligible. Inline statistics in DB2 11 also improve zIIP offload capability with up to 30% additional CPU offload to zIIP.

In addition to deferring the shadow index build (described above), the SORTNPSI utility keyword or the REORG_PART_SORT_NPSI subsystem parameter affect whether inline statistics will be collected for NPSIs during part level REORG.

Measurements on a UTS table with 100 million rows and 5 non-partition indexes showed that collecting histogram statistics up increases the CPU time for LOAD, REORG, and REBUILD by up to 40%. The results are shown in Figure 10-12.

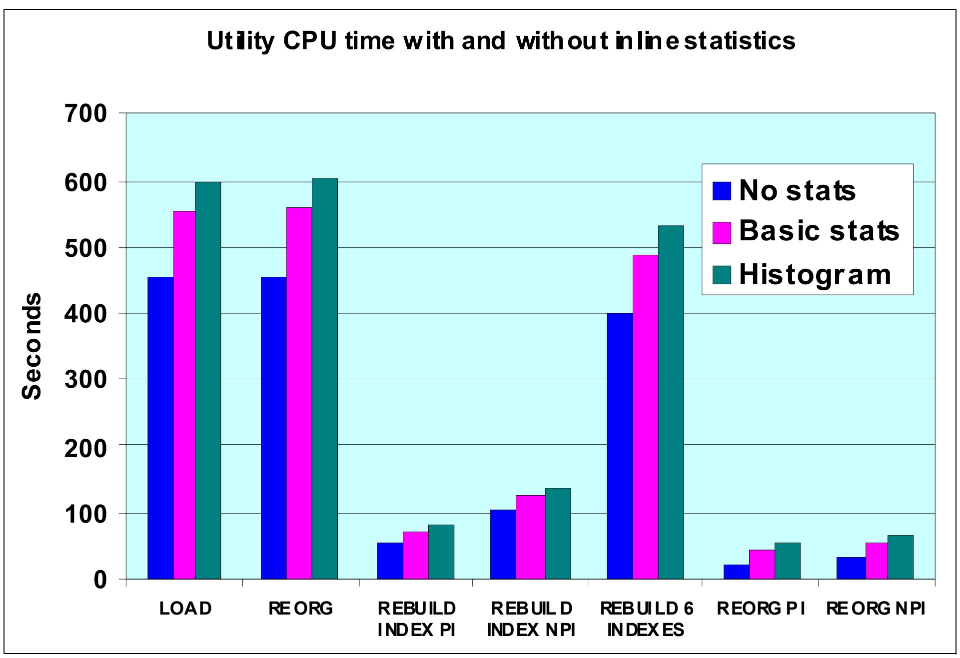

Figure 10-12 Utility CPU time with inline statistics

However, if we compare inline statistics to running RUNSTATS to gather these same statistics in addition to running LOAD, REORG, and REBUILD, the result is up to 40% elapsed time reduction and only a 7% CPU increase. The results are shown in Figure 10-13.

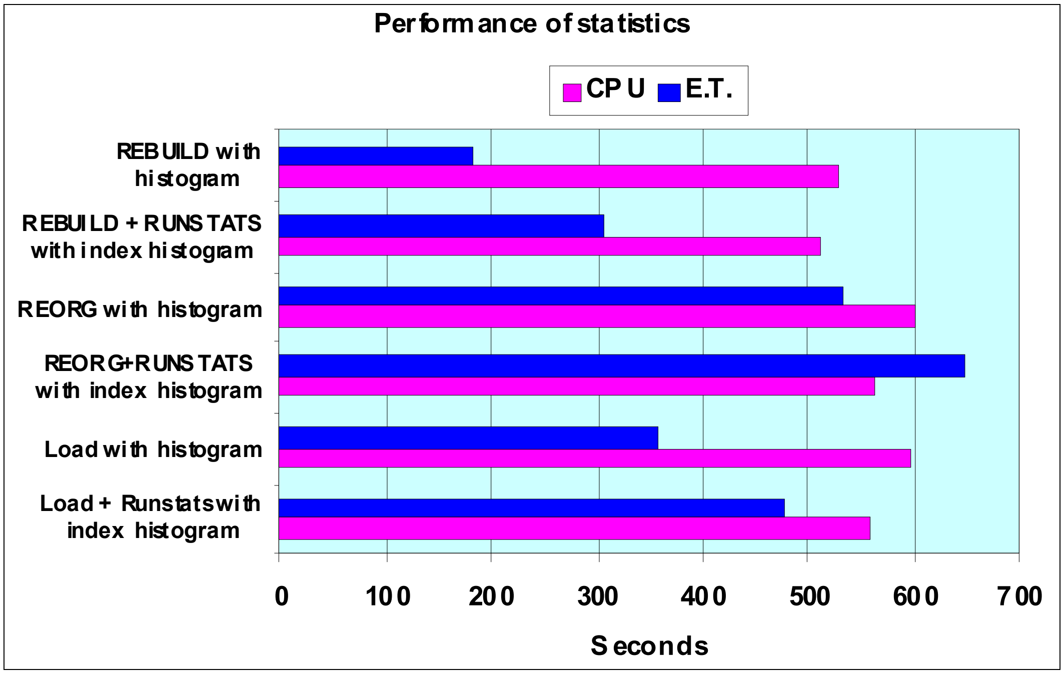

Figure 10-13 Impact of inline statistics

DB2 11 increases the amount of RUNSTATS CPU time that is eligible for zIIPs. Figure 10-14 illustrates an example of RUNSTATS with the COLGROUP keyword on three columns. Although DB2 11 increases the total CPU time by 8.4%, DB2 11 reduces the amount of non-zIIP-eligible time by 75%, thereby reducing the cost of ownership.

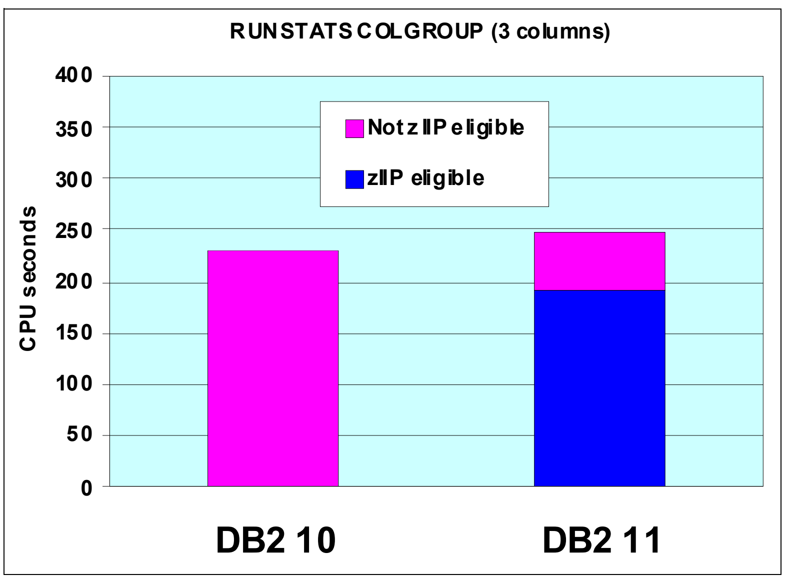

Figure 10-14 RUNSTATS COLGROUP

10.5 Partition-level image copies

Before DB2 11, when multiple partitions were REORG'd at the same time, REORG created a single inline image copy data set. Inline image copies can be too large to handle when reorganizing many partitions in a single REORG. Users require the ability to create inline image copies at the partition level for better data set manageability and recoverability. In DB2 11, REORG was changed to create partition-level inline copies. This is already the behavior for IBM FlashCopy® inline copies. This also simplifies recovery processing. Partition-level image copies are created if a valid TEMPLATE with &PA or &PART is specified on the REORG COPYDDN statement.

The change improves partition-level RECOVER performance from image copies created by partition Online REORG.

Performance tests were run on a four-way z196 processor, no zIIP, using a non-data sharing system against a UTS table with 100 million rows and 100 partitions with five NPIs showing significant improvement in RECOVER CPU and elapsed time. See Figure 10-15.

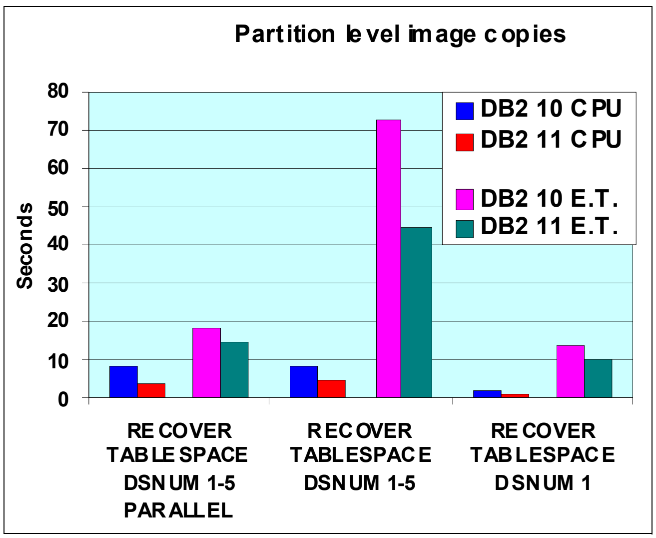

Figure 10-15 Partition level image copies

- Recover five partitions in parallel from image copies created after online REORG part(1:5) showed 52% CPU improvement and 21% elapsed time improvement.
- Recover five partitions serially from image copies created after online REORG part(1:5) showed 49% CPU improvement and 39% elapsed time improvement.
- Recover one partition from image copies created after online REORG part(1:5) showed 49% CPU improvement and 28% elapsed time improvement.

10.6 Cross Loader

The Cross Loader utility copies data from one table to another. When copying large objects (LOBs), DB2 10 used LOB locator variables, which must be acquired and freed for each LOB of each row. LOB locator variables are necessary when the SUBSTR function is used to read portions of a LOB in random sequence, but DB2 9 introduced FETCH CONTINUE to improve the performance when a LOB is read sequentially. Cross Loader in DB2 11 switched from LOB locator variables to FETCH CONTINUE.

DB2 11 reduced the class 2 CPU time for Cross Loader to copy 1 MB LOBs locally from one table to another by 28%. The elapsed time was unchanged.

DB2 11 is also expected to reduce the CPU time with small LOBs, perhaps even more than 28%. However, with small LOBs, the CPU benefit is diluted by the processing time for non-LOB columns.

10.7 Utility MRU

See 2.6.2, “MRU management” on page 43 on buffer management enhancements for utility using (most recently used) MRU.

10.8 Suppress NULL index support

See 8.1, “Suppress NULL key” on page 166 for utility support of suppressed NULL indexes.

11

Installation and migration

DB2 11 provides ease of migration by providing the option to keep application compatibility. Plan management access path reuse has been expanded to keep package migration easy. Catalogs and directories have less structure changes in DB2 11 compared to DB2 10, which contributes to a significantly faster migration performance.

This chapter contains the following topics:

- DB2 11 key requirements
- Catalog migration
- Conversion of catalog and directory tables to extended RBA format
- Improvement in access path reuse
- Application compatibility
- System parameters related with performance

IBM provides a no-charge migration workshop to help users plan for it. A starting point is the information that can be found at the external developers work open community at the following site:

https://ibm.biz/DB211MPW

11.1 DB2 11 key requirements

Before you begin the installation or migration process, look at the big picture. You need to be aware of the major requirements to get from your current version of DB2 to DB2 11 for z/OS. You need to know where you are currently and where you need to be before you embark on this process, which includes considerations not only on DB2 but also z/OS and tools.

Table 11-1 points out the most recent versions, program numbers, availability dates, minimum z/OS level required, and marketing and service termination dates.

Table 11-1 DB2 versions and required z/OS level

DB2 Version	Program number	Availability date	Minimum z/OS level	End of marketing	End of service
9	5635-DB2	March 2007	z/OS V1R7	**December 2010**	**June 2014**
10	5605-DB2	October 2010	z/OS V1R10	TBD	TBD
11	**5615-DB2**	**October 2013**	**z/OS V1R13**	TBD	TBD

Keep in mind that the required level of z/OS is the one at which the version of DB2 began to be developed. There might be functions that are only available with later levels of z/OS.

We summarize what z/OS 2.1 brings in 11.1.2, “DB2 related enhancements in z/OS 2.1” on page 249.

For DB2 10, you were able to get to it via a skip-level migration from DB2 V8. For DB2 11, skipping level is not allowed. This means that if you are on DB2 9, you first must move to DB2 10 to migrate to DB2 11. As a result, the migration process has fewer modes than we saw for the migration to DB2 10.

The modes that you can use while migrating to DB2 10 are DB2 11 conversion mode (CM), DB2 11 CM*, DB2 11 enabling-new-function mode (ENFM), DB2 11 ENFM*, DB2 11 NFM. Figure 11-1 on page 247 summarizes the possible migration modes and how you can get to each one of them.

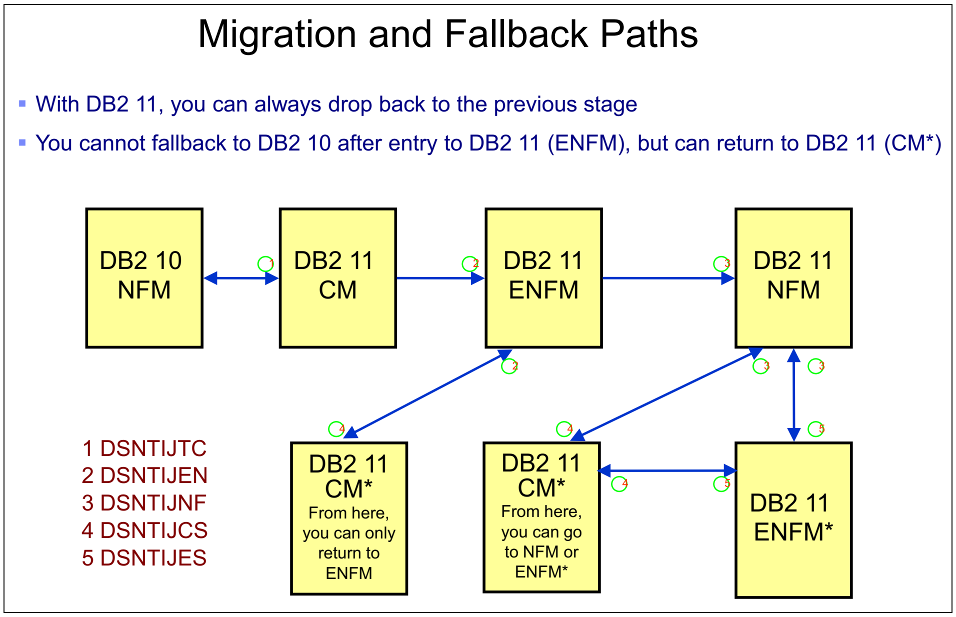

Figure 11-1 Migration and fallback paths with DB2 11

11.1.1 Operating system and support programs

DB2 11 requires the function that is provided by the following licensed programs or their equivalents. Subsequent versions or releases of these products are acceptable:

- z/OS V1.13 Base Services (5694-A01) with the following base and optional elements:
  - DFSMS V1.13
  - IBM Language Environment® Base Services
  - z/OS V1.13 Security Server (RACF)
- IRLM V2.3 (delivered with DB2 11), plus APARs PM84765 and PM85053

Processors

DB2 11 for z/OS operates on z10 or higher processors

Transaction managers

- Information Management System (IMS)
  - IMS V13 (5635-A04)
  - IMS V12 (5635-A03)
  - IMS V11 (5635-A02)
- Customer Information Control System (CICS)
  - CICS Transaction Server for z/OS, V5.1 (5655-Y04)
  - CICS Transaction Server for z/OS, V4.1 and V4.2 (5655-S97), or later
  - CICS Transaction Server for z/OS, V3.1 and V3.2 (5655-M15), or later

DRDA connectivity

DB2 for z/OS supports Distributed Relational Database Architecture (DRDA) as an open interface, allowing access from any client. If you want a seamless migration where your remote applications must continue to operate throughout the migration process, DB2 Connect Version 10.1 Fixpack 2 or DB2 Connect Version 9.7 Fixpack 6, or later, are recommended clients to help support such a seamless migration when you migrate your DB2 data sharing group.

DB2 Connect Version 10.5 Fixpack 2 uses DB2 11 features. All versions can access DB2 11 for z/OS but Version 10.5 Fixpack 2 or higher is needed to use some of the DB2 11 components such as the following features:

- Array support
- Autocommit performance improvements for procedures and cursors
- Data sharing support for global variables
- Longer client information fields

DB2 11 acting as a client supports the following relational database products:

- DB2 for Linux, UNIX, Windows 9.5 (5765-F41), or later
- DB2 Enterprise Server (ESE) for Linux, UNIX, and Windows V9.5 (5765-F41), or later
- DB2 Express Edition for Linux, UNIX, and Windows, V9.5 (5724-E49), or later
- Database Enterprise Developer Edition V9.5 (5724-N76), or later
- DB2 for IBM iSeries® V6.1 (5761-SS1), or later
- DB2 Server for VSE and VM V7.3 (5697-F42), or later
- Any other DRDA compliant relational DBMS server

Web connectivity

It is provided by any of the DB2 Connect clients using one of the IBM Data Server clients or drivers.

JDBC: DB2 11 supports the following JDBC application programming interface (API) specification levels:

- JDBC 3.0 API requires any of the following at run time:
  - IBM 31-bit SDK for z/OS, Java 2 Technology Edition, V5 (SDK5) (5655-N98), or later
  - IBM 64-bit SDK for z/OS, Java 2 Technology Edition, V5 (SDK5) (5655-N99), or later
- JDBC 4.0 API requires any of the following at run time:
  - IBM 31-bit SDK for z/OS, Java Technology Edition, V6 (SDK6) (5655-R31), or later
  - IBM 64-bit SDK for z/OS, Java Technology Edition, V6 (SDK6) (5655-R32), or later

DB2 components

Figure 11-2 summarizes the components of DB2 11 available in its packaging.

For details, see the DB2 for z/OS Information Roadmap at the following site:

http://www.ibm.com/support/docview.wss?uid=swg27039818

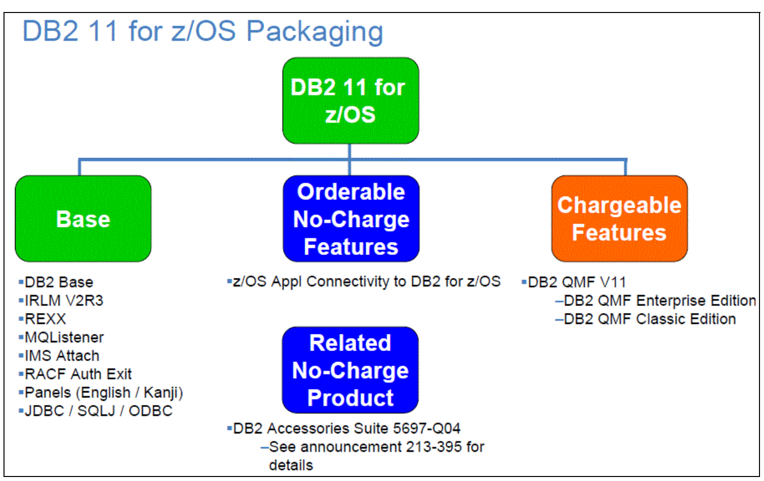

Figure 11-2 DB2 packaging

IBM Tools for Performance Management include the following:

- DB2 Query Monitor for z/OS, V3.2.0 (5655-V42), or V3.1 plus APAR PM75732
- DB2 SQL Performance Analyzer for z/OS, V4.2.0 (5655-W60) plus APAR PM97648 or V4.1, plus APAR PM59925
- IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS, V5.2 (5655-W37)
- InfoSphere OptimQuery Workload Tuner for DB2 for z/OS, V4.1 (5655-AA4)

For details about tools support, see the following site:

http://www-01.ibm.com/support/docview.wss?uid=swg21609691

11.1.2 DB2 related enhancements in z/OS 2.1

This section summarizes the DB2 11 functions that require z/OS 2.1:

> **Important:** Some DB2 11 functions require z/OS 2.1 rather than the minimum required level z/OS 1.13. If there are functions you need with z/OS 2.1, you need to take into account the z/OS migration first.

- Enhancing DB2 BACKUP SYSTEM solution (DB2 11 only)
  - Enable recovery of single pageset from DB2 system-level backup even if original volume does not have sufficient space
  - Enable exploitation of FlashCopy consistency group for DB2 BACKUP SYSTEM
  - Enable restore of a pageset to a different name
- z/OS DFSMS VSAM RLS for Catalog support (DB2 9 and above)

  Improved DB2 data set open/close performance
- z/OS DFSMS Storage Tiers (DB2 10 and above)
  - Optimizes disk placement on SSD and HDD
  - DB2 10 and 11 provide the ability to issue STOP DB AT COMMIT synchronously
- Enhanced WLM managed buffer pools (DB2 9 and above)

  WLM now allows buffer pool sizes to shrink
- XES GBP write-around support (DB2 11 only)

  CFLEVEL 17 or above (retrofit to z/OS 1.13 with OA37550)
- Reduced DRDA message latency and DB2 CPU reduction (DB2 11 only)

  TCP/IP synchronous receive (retrofit to z/OS 1.13 with OA39810) and PM80004
- Reduced DRDA message latency and DB2 CPU reduction (DB2 9 and above)
  - CommServer SMC-R support for RDMA over Converged Ethernet (RoCE), Network Adapter
  - Requires zEC12 or zBC12, z/OS to z/OS only for now
- 2 GB page frame support (DB2 11 only)
  - Requires zEC12 or zBC12
  - Retrofit to z/OS 1.13 with APAR OA40967 and z/OS V1R13 RSM Enablement Offering

- Pageable 1 M frame support (DB2 10 and 11)
  - Requires zEC12 or zBC12 (retrofit to z/OS 1.13)
  - DB2 10 APAR PM85944
- CPU reduction for Media Manager I/O (DB2 9 and above)
- Ability to execute code loaded into 1 MB pageable frames (DB2 11 only and Flash Express configured)

  SVL measured 1.8% CPU reduction for an OLTP workload

11.2 Catalog migration

DB2 11 is upwardly compatible with earlier releases of DB2 for z/OS. Migration with full fallback protection is available for customers running on DB2 10 for z/OS. Existing customers should ensure that they are successfully running on DB2 10 for z/OS (NFM) before migrating to DB2 11. For more migration information, see Info APAR II14660: DB2 10 MIGRATION/FALLBACK INFOAPAR TO/FROM DB2 11.

Before migration, apply fallback SPE PM31841, and standard cleanup of SYSLGRNX and SYSCOPY plus checking SYSUTILX for pending actions should be performed.

During migration, be aware of the functions that are deprecated in DB2 11. Although they are supported in DB2 11, support for these functions might be removed in the future. Avoid creating new dependencies that rely on these functions, and if you have existing dependencies on them, develop plans to remove these dependencies. Refer to *DB2 11 for z/OS Technical Overview,* SG24810 for details.

There is no skip migration support with DB2 11 and fallback can be done to DB2 10.

11.2.1 Summary of catalog and directory changes

In DB2 V8, DB2 catalogs and directory were converted to Unicode. In DB2 10, there were table space organization conversions to UTS. Although there are as many tables and indexes updated through DB2 11 migration processing, there are fewer table space conversions in DB2 11 compared to DB2 10.

Table 11-2 shows the progression of the number of objects of the DB2 catalog and directory across the DB2 versions.

Table 11-2 Number of catalog and directory objects

Version	Table spaces	Tables	Indexes	Columns	LOB columns	Inline LOB columns
V1	11	25	27	269	0	0
V3	11	43	44	584	0	0
V5	12	54	62	731	0	0
V7	20	84	118	1212	2	0
V8	22	85	132	1265	2	0
V9	28	104	165	1652	6	0

Version	Table spaces	Tables	Indexes	Columns	LOB columns	Inline LOB columns
V10	95	134	233	2036	36	4
V11	108	143	250	2202	42	9

During the migration to DB2 11 CM, some table spaces, tables, and indexes are created as well as new columns are added. During ENFM conversion processing, table spaces are converted to universal table spaces, partitioned by growth organization. This triggers six online REORG steps during ENFM.

DB2 11 adds one more optional step to convert log RBA of catalog and directory to extended format. This step can be done any time in NFM.

11.2.2 Migration steps and performance

The migration path to DB2 11 continues to use the same approach as with DB2 10. Figure 11-3. Figure 11-3 illustrates the migration paths and the resulting migration modes.

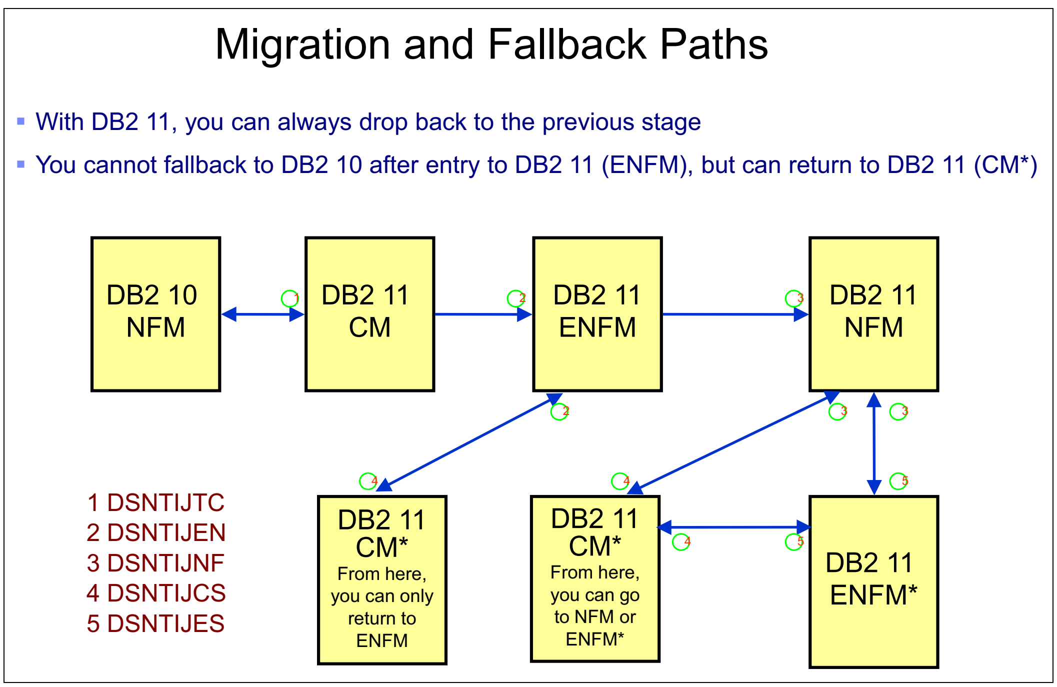

Figure 11-3 Migration path and modes

DB2 11 supports the following modes:

- CM: conversion mode

  The mode DB2 is in when DB2 11 is started for the first time after migrating direct from DB2 10. It will still be in CM when migration job DSNTIJTC has completed. Very little new function can be executed in CM. Data sharing systems can have DB2 10 and DB2 11 members in this mode.

- ENFM: enabling-new-function mode

  This mode is entered when job DSNTIJEN is first executed (CATENFM START). DB2 remains in this mode until all the enabling functions are completed. Data sharing systems can only have DB2 11 members in this mode.

- NFM: new function mode

  This mode is entered when job DSNTIJNF executed (CATENFM COMPLETE). This mode indicates that all catalog changes are complete and ready to use new functions.

- ENFM*

  This is the same as ENFM but the * indicates that at one time DB2 was at DB2 11 NFM. Objects that were created when the system was at NFM can still be accessed but no new objects can be created. When the system is in ENFM*, it cannot fallback to DB2 10 or coexist with a DB2 10 system.

- CM*

  This is the same as CM but the * indicates that at one time DB2 was at a higher level. Objects that were created at the higher level can still be accessed. When DB2 is in CM*, it cannot fallback to DB2 10 or coexist with a DB2 10 system.

Restrictions in conversion mode

Although most of the performance features are available in CM, some of new features added in DB2 11 are not available to use as they preclude the fallback to DB2 10. Following are a few examples:

- EXCLUDE NULL Index
- DGTT NOT logged option
- Application compatibility (APPLCOMPAT) cannot be set as V11R1 in CM, or ignored if it is set as V11R1
- Extended RBA/LRSN conversion cannot be executed during CM

As a performance improvement in DB2 11, log records are written without the need to first space switch to the xxxxMSTR address space. To support this change, log buffers must be moved from their current location in xxxxMSTR 31-bit private to common storage. Since the log buffers can be large, up to 400 MB in size, it is not practical to move the log buffers to ECSA because most systems would not have enough ECSA available for a single request of this size. The log buffers are moved to 64-bit common (HCSA).

The amount of HCSA used is roughly the size of the log buffers specified by the OUTBUFF parameter plus 15% (for control blocks). The SYS1.PARMLIB setting for HVCOMMON must be large enough to accommodate this size for each DB2 11 subsystem active on an LPAR. In addition, the buffers can reside in 1 MB page frames, if available. You might want to increase the SYS1.PARMLIB setting for LFAREA to allow for this allocation.

Important: Ensure that the IEASYSxx HVCOMMON has enough space for the log buffers.

Considerations in new function mode

There are three important steps in NFM for DB2 11 to fully enable the extended RBA/LRSN format:

- Convert the BSDS records to support extended format (install job DSNTIJCB). This causes outage for non-data sharing subsystems since you need to plan for stopping DB2 and run the DSNTIJCB. It is a very fast execution. In data sharing, you can convert a member at the time, and avoid outage.

- Convert the DB2 catalog and directory table spaces and indexes to extended format (install job DSNTIJCV). It is mainly a REORG SHRLEVEL CHANGE. It causes no outage. The DSNTIJCV job could be broken up by object and executed a bit at the time if necessary.
- Convert the user data.

Important: When applying the conversion to extended RBA/LRSN, consider allowing space for some extra LOG allocation.

11.2.3 Catalog migration in non-data sharing

To understand the migration performance, we selected two customers provided DB2 catalogs of different sizes:

- Customer 1 (C1) catalog size 23.8 GB
- Customer 2 (C2) catalog size 15.2 GB

They are migrated from DB2 9 NFM to DB2 10 CM, then DB2 10 NFM as a base measurement. Then, we proceed to the DB2 10 to DB2 11 migration to CM, then new function mode.

The measurement environment consisted of the following components:

- One dedicated z/EC12 LPAR with 4 processors and 40 GB real storage
- z/OS 1.13
- DS8800 disk controller
- Buffer pool sizes are defined as below for all of the measurement steps:

  BP0=20 K, BP4=10 K, BP8K=50 K, BP16=10 K, BP32K=50 K

Measurement steps

Both CM migration and NFM conversions are measured as two-step measurements under the same configurations:

- CM migration: Execute DSNTIJTC (CATMAINT UPDATE) to migrate the catalogs:
  - DB2 9 NFM to DB2 10 CM
  - DB2 10 NFM to DB2 11 CM
- Enabling new function: Execute DSNTIJEN and DSNTIJNF, which include steps of online REORG TABLESPACE:
  - DB2 10 CM to DB2 10 NFM

    Online REORG 9 table spaces: SPT01, DBD01, SYSDBASE, SYSDBAUT, SYSGROUP, SYSOBJ, SYSPKAGE, SYSPLAN, SYSVIEWS
  - DB2 11 CM to DB2 11 NFM

    Online REORG 6 table spaces: SYSLGRNX (SHRLEVEL CHANGE), SYSCOPY, SYSRTSTS, SYSTSIXS (SHRLEVEL CHANGE), SYSSTR, SYSTSTAB (SHRLEVEL CHANGE), new columns, new indexes, some new SMS controlled table spaces and indexes.

    Objects with functional dependencies are indicated by “P” in the IBMREQD catalog column.

Table 11-3 summarizes the results of the migration steps for C1.

Table 11-3 Migration comparison of catalog C1 (time in seconds)

Catalog 1 (23.8 GB)	DB 9 to DB2 10		DB2 10 to DB2 11	
Non-data sharing	**CPU**	**Elapsed**	**CPU**	**Elapsed**
Migration to compatibility mode (CATMAINT UPDATE)	28.77	124	0.07	7.67
Migration to NFM (Online REORG)	911	1660	62	92
Total	939	1784	62	100

With the 23.8 GB user catalog, by comparing the two steps, you can see a 16 times elapsed time improvement in step1, migrating from DB2 10 NFM to DB2 11 CM, compared to DB2 9 NFM to DB2 10 CM migration. To convert to NFM (step 2), 18 times less elapsed time is seen in DB2 11 compared to DB2 10.

The migration process from DB2 10 CM to DB2 10 NFM was much slower in CPU and elapsed time than the migration process from DB2 11 CM to DB2 11 NFM because a much larger number of catalog and directory tables and total rows were processed in DB2 10 than in DB2 11. See also Table 11-2 on page 250.

Table 11-4 summarizes the results of the migration steps for C2.

Table 11-4 Migration comparison of catalog C2 (time in seconds)

Catalog 2 (15.2 GB)	DB2 9 TO DB2 10		DB2 10 TO DB2 11	
Non data sharing	**CPU**	**Elapsed**	**CPU**	**Elapsed**
Migration to compatibility mode (CATMAINT UPDATE)	4.38	14.0	0.08	8.99
Migration to NFM (Online REORG)	303	619	7.34	26.6
Total	307	633	7	36

Using the 15.2 GB user catalog, the elapsed time reduction in step 1 and step 2 shows a 17 times reduction in DB2 11 compared to DB2 10 migration.

11.2.4 Catalog migration in data sharing

The same customer 1 catalog is migrated from DB2 10 NFM to DB2 11 CM then NFM in a two-way data sharing system with no activity in the other member to understand the data sharing impact from V10 NFM to V11CM then V11NFM.

The performance of the migration of the 23.8 GB customer catalog in the data sharing environment was similar to the non-data sharing environment.

11.2.5 Conclusions and considerations

Important: Catalog migration performance from DB2 10 to DB2 11 is significantly faster than migration from DB2 9 to DB2 10.

This is because there are fewer changes to the catalog and directory and fewer rows are updated by DB2 11 migration processing. Especially SPT01 tends to be very large because it contains package and statement information.

During DB2 10 migration, online REORG of SPT01 is required while *REORG of SPT01 is not included in DB2 11 migration*. In general, we find having *large buffer pools* for the catalog and directory help the migration performance by reducing the synchronous I/Os against catalog and directory tables and indexes.

11.3 Conversion of catalog and directory tables to extended RBA format

This section describes conversion to extended relative byte address/log record sequence number (RBA/LRSN) format for DB2 catalog and directory.

11.3.1 Catalog and directory conversion

When DB2 11 is in NFM, convert DB2 catalog and directory to the extended RBA format:

- For new installations, all of the catalog and directory table spaces and index spaces are created as extended log format and there is need of further action only for the BSDS conversion.
- For migrated subsystems, DB2 provides a new installation job DSNTIJCV to convert the format of table space and indexes on catalog and directory. The job performs COPY and REORG against the catalog and directory.

The conversion to extended RBA/LRSN is not required if the subsystem is not reaching to log RBA/LRSN limit.

11.3.2 Measurement result of DSNTIJCV

Performance evaluation on DSNTIJCV is done using the same customer catalog 1 (23.8 GB size) described above. This measurement was done in a non-data sharing system with the same configuration as measurement of catalog migration of the customer 1 catalog.

Conversion of catalog and directory tables to extended format in DB2 11 NFM shows elapsed time of 9 minutes and 46 seconds and CPU time of 4 minutes and 2 seconds. The performance of the conversion is highly dependent on the size of tables and indexes. Standard REORG recommendations apply: MODIFY RECOVERY to remove obsolete SYSCOPY and SYSLGRNX recovery records, and larger buffer pools to speed up the process. The job can be divided in steps and is concurrent with normal activity.

11.4 Improvement in access path reuse

DB2 access plan management was introduced in DB2 9 for z/OS as a way of providing a fall-back option in case of access path degradation during REBIND.

With plan management, DB2 saves old copies of packages at REBIND time, so you can SWITCH back to previous copies in case of problems. Plan management can be used in two modes:

- BASIC: DB2 keeps the current and the previous copies of the package
- EXTENDED: DB2 where you keep three copies: the current, the previous, and the original one of the packages.

REBIND SWITCH allows you to decide to which copy to fall-back to.

The default in DB2 11 is EXTENDED. Figure 11-4 shows an overview of how this option works.

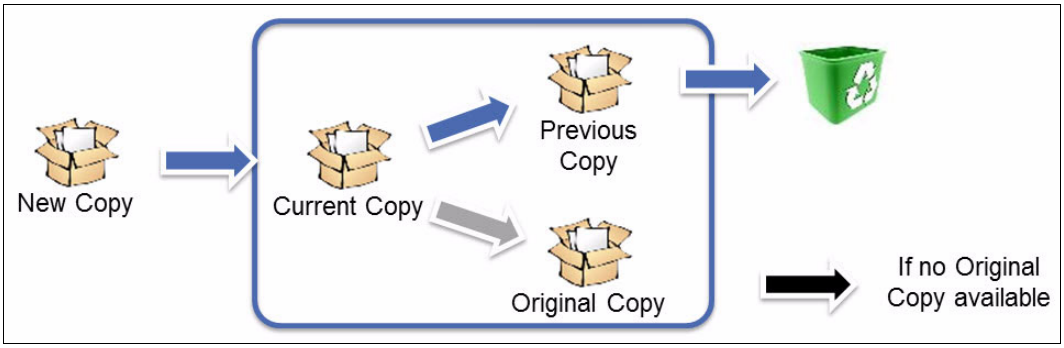

Figure 11-4 Plan management overview

With DB2 plan management EXTENDED, you can fall-back to the previous or to the original copy of a package.

Each one of the three packages are maintained independently. Dropping an object on which package copies depend on will impact each copy in consequence and, based on the extent of the object-package relationship, one or more may become non-operative. This is important to understand before dropping indexes or views.

The catalog table SYSIBM.SYSPACKCOPY contains SYSPACKAGE-like information about previous and original copies of packages. By using this table, you can verify the status and characteristics of previous packages copies before doing a REBIND SWITCH.

The query in Example 11-1 combines the SYSPACKAGE and SYSPACKCOPY information to get a view of all the copies available for a package.

Example 11-1 Query on SYSPACKAGE and SYSPACKCOPY

```
SELECT
SUBSTR(COLLID,1,10) AS COLLID ,SUBSTR(NAME,1,10) AS NAME ,LASTUSED
,VALID ,OPERATIVE ,COPYID ,PLANMGMT ,APRETAINDUP
FROM SYSIBM.SYSPACKAGE
WHERE COLLID = 'CRISCOLL' AND NAME = 'DSN8CLTC'
UNION ALL
SELECT
SUBSTR(COLLID,1,10) AS COLLID ,SUBSTR(NAME,1,10) AS NAME ,LASTUSED
,VALID ,OPERATIVE ,COPYID ,PLANMGMT ,APRETAINDUP
FROM SYSIBM.SYSPACKCOPY
WHERE COLLID = 'CRISCOLL' AND NAME = 'DSN8CLTC'
WITH UR;
```

Example 11-2 shows the output of this query.

Example 11-2 Query on SYSPACKAGE and SYSPACKCOPY output example

```
---------+---------+---------+---------+---------+---------+---------+----
COLLID    NAME     LASTUSED   VALID OPERATIVE COPYID PLANMGMT APRETAINDUP
```

```
---------+---------+---------+---------+---------+---------+---------+----
CRISCOLL DSN8CLTC 2012-12-28 Y     Y         0      E        Y
CRISCOLL DSN8CLTC 2012-07-27 Y     Y         1      E        Y
CRISCOLL DSN8CLTC 2012-03-01 Y     Y         2      E        Y
DSNE610I NUMBER OF ROWS DISPLAYED IS 3
```

The LASTUSED column of the SYSIBM.SYSINDEXSPACESTATS table allows you to identify an index that has not been used for a long time. But before deciding to remove an index, think about the possibility of making a package's old copy non-operative if there is a dependency on that index.

The SWITCH option allows you to revert to an older copy of a package in the event of a performance regression. You cannot specify SWITCH with any other rebind options. If the package that you specify does not have the previous or original copy to switch to, DB2 issues an error message, and continues processing the rest of the list. Following are the accepted values:

- PREVIOUS: Switches between the current and previous copies of a package. The current copy takes the place of the previous copy, and the previous copy takes the place of the current copy.
- ORIGINAL: Moves the current copy to take the place of the previous copy. Any previous copy that exists is purged. The original copy is cloned to take the place of the current copy.

Figure 11-5 illustrates the SWITCH original behavior.

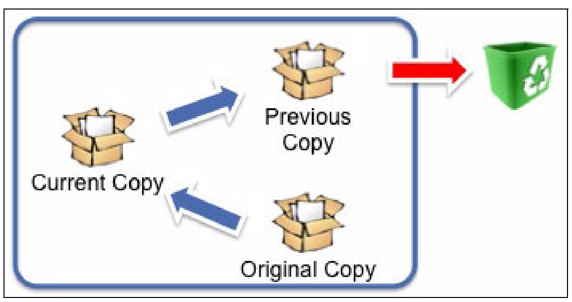

Figure 11-5 SWITCH original copy

Figure 11-6 illustrates an example of the SWITCH previous command effects.

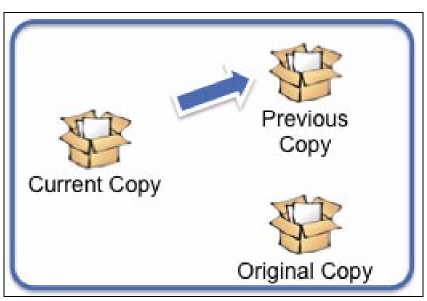

Figure 11-6 SWITCH previous copy

11.4.1 APREUSE BIND/REBIND option

The APREUSE option in BIND or REBIND specifies whether DB2 tries to reuse previous access paths for SQL statements in a package. DB2 uses information about the previous access paths from the directory to create a hint.

When migrating to a new DB2 version or release, the APREUSE REBIND option can be used to ensure that the static SQL statements in a package reuse the existing access paths, but at the same time generates new run time structures. This provides access path stability across REBINDs in the migration processing.

In DB2 10, the APREUSE option was only possible when all the statements in a package were able to reuse the existing access path. It was all or nothing. Under the APREUSE (ERROR) bind option, the rebind operation fails for a package after all of the statements are processed whenever any access paths cannot be reused.

The following valid values apply for APREUSE:

- NONE: DB2 does not try to reuse previous access paths for statements in the package.
- NO: An accepted synonym for NONE.
- ERROR: DB2 tries to reuse the previous access paths for SQL statements in the package. If statements in the package cannot reuse the previous access path, the bind operation for the package that contains the statement ends, and processing continues for the next package. DB2 indicates the number of statements that cannot be reused in any package in a message. New and changed statements in a package never prevent the completion of the bind or rebind operation, even though no previous access path is available for reuse.
- WARN: DB2 tries to reuse the previous access paths for SQL statements in the package. DB2 ignores the previous access path and generates a new access path for any statement in the package that cannot reuse the previous access. The bind operation for the package completes, regardless of whether some access paths cannot be reused. DB2 indicates the number of statements that cannot be reused in any package in a message.

11.4.2 APREUSE(WARN)

Under the APREUSE(ERROR) bind option, the rebind operation fails for a package after all of the statements are processed whenever any access paths cannot be reused. In DB2 11, APREUSE(WARN) is introduced to circumvent this issue by reusing existing access paths whenever possible.

With APREUSE(WARN), DB2 tries to enforce the existing access path. If any of the statements cannot reuse the access path, DB2 creates an entirely new access path for that statement and completes the bind operation. At the end, DB2 will output a message indicating the number of statements that cannot reuse the previous access path.

While APREUSE(ERROR) option works at a package level, APREUSE (WARN) option works at the statement level. APREUSE (WARN) tries to reuse the existing access path for all the statements in a package, but when it is not possible, a new access path will be created for those statements. DB2 does so by extracting the access path specifications stored with the package (in the directory), and passing them to the optimizer as hints. This hint may or may not be applied. If DB2 cannot apply a hint for a statement in a package, the entire hint is discarded, and DB2 reoptimizes that statement from scratch without a hint.

This function is effective in DB2 11 CM.

Usage considerations

The `REBIND PACKAGE` command is preferred when access path changes are acceptable with APREUSE (WARN) because it enables the use of plan management policies. This approach allows you to switch the package to use the previous access paths when you detect access path changes that you do not want.

APREUSE (WARN) along with APCOMPARE (WARN) can be used for access path comparison. It tries to enforce the reuse of access paths, accept access path changes when reuse cannot be applied, and find out when changes occur. This option might result in unacceptable access path changes. Therefore, the recommendation is to also use a plan management policy that enables you to switch to previous access paths. If all the changes are acceptable, REBIND with APREUSE (WARN). It uses less DB2 CPU time compared to REBIND without APREUSE option.

When APREUSE (WARN) is used along with EXPLAIN (YES), DB2 populates PLAN_TABLE rows with information about the access paths that changed as a result of the rebind operation. You can use the PLAN_TABLE data to identify and analyze the changed access paths.

When APREUSE is used along with EXPLAIN (ONLY), the bind and rebind operations are not completed. However, DB2 processes the access paths and populates PLAN_TABLE with access path result information. When reuse succeeds, DB2 inserts the value 'APREUSE' into the PLAN_TABLE.HINT_USED column. When reuse or comparison fails, DB2 populates the PLAN_TABLE.REMARKS column with information about the failures and the new access path. Be aware that EXPLAIN (ONLY) with APREUSE consumes more CPU than just using APREUSE because of having to go through the optimizer twice.

When both APREUSE and explicit optimizer hints are in effect for a statement, optimizer hints take precedence and APREUSE is ignored for that statement.

11.4.3 Performance measurements

The following performance measurements are presented to demonstrate the CPU usage variation when the APREUSE feature is used with different settings for EXPLAIN in DB2 11. These results are based on executing a REBIND of 100 packages containing 100 statements.

11.4.4 Rebind with APREUSE and EXPLAIN

This set of tests rebinds 100 packages each containing 100 statements with and without APREUSE(WARN) for different EXPLAIN settings.

Figure 11-7 demonstrates the potential cost of the APREUSE(WARN) option using an extreme case (100% warnings) as examples.

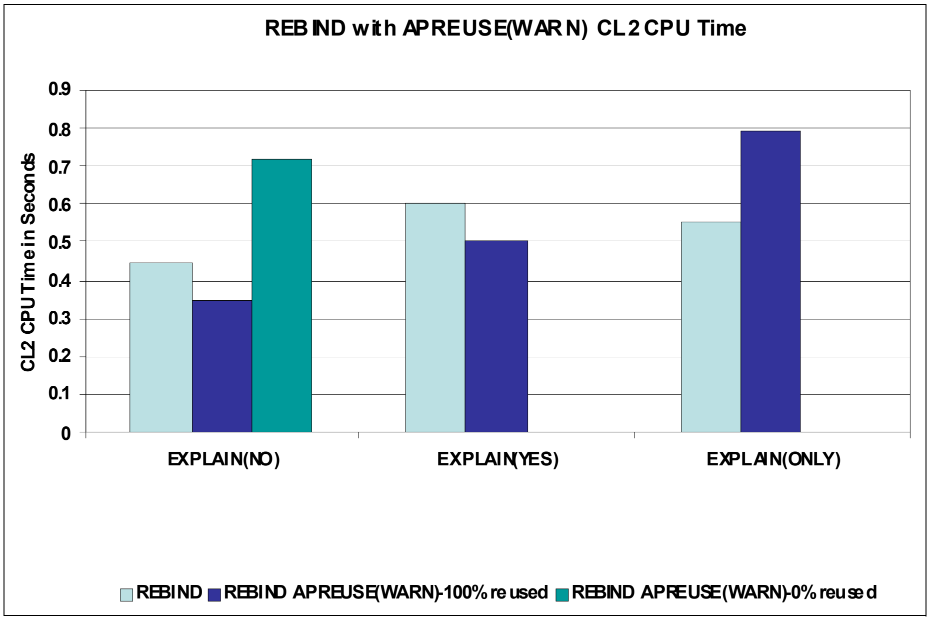

Figure 11-7 DB2 11 REBIND with APREUSE/EXPLAIN

The first bar of each chart set represents the base CPU time, which is the REBIND operation of 100 packages without APREUSE option.

The first set shows REBIND with EXPLAIN(NO).

The second bar is REBIND with APREUSE(WARN) where all statements in all packages were able to reuse the access path.

When the access path can be reused, REBIND with APREUSE uses less CPU time than without reusing the access path. This is because DB2 can skip a portion of access path selection processing.

The third bar represents REBIND with APREUSE(WARN) where all statements in all packages were unable to reuse access path.

REBIND with EXPLAIN(YES) causes a slight CPU increase due to additional EXPLAIN processing.

However, EXPLAIN(ONLY) case is different. EXPLAIN (ONLY) requires additional processing in the optimizer, which causes a CPU increase compared to REBIND with APREUSE.

11.5 Application compatibility

DB2 11 for z/OS has mechanisms in place to limit potential SQL and XML incompatibilities on application statements by allowing you to do the following functions:

- Identify applications affected by incompatible SQL and XML changes through trace records. This function provides a mechanism to discover which applications are affected:
  - DSNTIJPM migration job updated to warn of static SQL packages affected
  - IFCID 366 and 376 updated to report on dynamic SQL and static

- Control the compatibility level to DB2 10 at application level. Following are the methods to make the transition to new behavior:
  - APPLCOMPAT(VnnR1) BIND/REBIND option for static SQL
  - APPLCOMPAT subsystem parameter for static SQL – indicates default value
  - CURRENT APPLICATION COMPATIBILITY special register for dynamic SQL
  - DSN_PROFILE_ATTRIBUTES for DRDA applications

For dynamic SQL statements, the APPLICATION COMPATIBILITY special register value must be set to an appropriate value and for static SQL statements the APPLCOMPAT bind option must contain the wanted value.

APPLCOMPAT applies only to DML (or DDL that contains DML, such as CREATE VIEW, CREATE MASK, MQTs) not DDL nor DCL. For instance, DB2 allows the user to CREATE objects but application compatibility will not be checked until the object is referenced:

- CREATE TYPE (array) is allowed but SET array-variable = …. would be subject to application compatibility rules (that is, array-variable only available in V11R1 mode)
- BIND option applies to packages: stored procedures, UDFs, triggers, applications, and so on

APPLCOMPAT parameters

Although DB2 tries to avoid the incompatible changes as much as possible, sometimes incompatible changes cannot be avoided to comply SQL standards. DB2 11 application compatibility support helps you decide when your applications are ready for new SQL functionality.

APPLCOMPAT BIND/REBIND option for static SQL and CURRENT APPLICATION COMPATIBILITY special register for dynamic SQL can dictate new SQL and XML behaviors with incompatibilities. The system parameter APPLCOMPAT can be used to define the default values.

Acceptable values for the APPLCOMPAT parameter or special register are V10R1 and V11R1. When you migrate a DB2 subsystem, the system parameter APPLCOMPAT is set V10R1. When you install a new DB2 subsystem, it defaults to V11R1:

- V10R1 The default BIND option is V10R1 compatibility behavior. This is applicable in CM and new function mode.
- V11R1 The default BIND option is V11R1 compatibility behavior. This value is only allowed in new function mode.

Addressing incompatibility

APPLCOMPAT provides ease of migration by allowing the user to defer the action against the incompatibility issues. However, our suggestion is to address incompatibility issues while in DB2 11. DB2 plans to support the older SQL compatibility for next two releases. For example, in DB2 11 plus one release, APPLCOMPAT = V10R1 is valid but not anymore with DB2 11 plus two releases.

To identify the packages with incompatibility, new IFCID 376 is added in addition to IFCID 366. IFCID 376 is a roll-up of 366 records. With both 366 and 376 turned on, DB2 11 writes one 376 record for each unique static or dynamic statement that is affected by incompatible changes.

11.6 System parameters related with performance

The following performance-related DB2 system parameters are newly added or updated in DB2 11.

11.6.1 New system parameters related to performance

Several system parameters are new and can affect performance.

INDEX_CLEANUP_THREADS

INDEX_CLEANUP_THREADS indicates the maximum number of threads that can be created to process the cleanup of pseudo-deleted index entries on this subsystem or data sharing member. Pseudo-deleted entries in an index are entries that are logically deleted but still physically present in the index.

Acceptable values are in the range 0 - 128, with a default of 10. The recommendation is to keep the default unless you know better.

See 4.2, “Pseudo-deleted index key cleanup” on page 83 for details.

MAXSORT_IN_MEMORY

MAXSORT_IN_MEMORY specifies the maximum allocation of in-memory sort storage in kilobytes for a query that contains an ORDER BY clause, a GROUP BY clause, or both.

Acceptable values: 1000 to the value specified in SORT POOL SIZE, whichever is larger, thus 1000 - 128000. The default value is 1000.

The storage is allocated only during the processing of the query. Increasing the value in this field can improve performance of queries with sort operations but might require large amounts of real storage when several such queries run simultaneously.

The recommendation is to start with the default value of 1000 because this is similar behavior as DB2 10, and to increase the value only if there are sort intensive queries that can take advantage of more in memory and there is enough real storage to support the concurrency.

PARAMDEG_DPSI

The PARAMDEG_DPSI system parameter specifies the maximum degree of parallelism that you can specify for a parallel group in which a data-partitioned secondary index (DPSI) is used to drive parallelism.

Following are the acceptable values:

- 0: Specifies that DB2 uses the value that is specified for the PARAMDEG subsystem parameter, instead of PARAMDEG_DPSI, to control the degree of parallelism when DPSI is used to drive parallelism. This is the default.
- 1: Specifies that DB2 creates multiple child tasks but works on one task at a time when DPSI is used to drive parallelism.
- 2 - 254: Specifies that DB2 creates multiple child tasks and works concurrently on the tasks that are specified. The number of specified tasks can be larger or smaller than the number of tasks as specified in PARAMDEG. When PARAMDEG is set to 1, the rest of the query does not have any parallelism.
- DISABLE: Specifies that DB2 does not use DPSI to drive parallelism. Parallelism might still occur for the query if PARAMDEG is greater than 1.

This subsystem parameter allows users to improve DPSI performance with parallelism, without increasing the number of parallel degrees generated for general query parallelism. Customers who currently use parallelism may choose to set PARAMDEG_DPSI to the default of 0 so that the degree is aligned with that chosen for regular query parallelism (PARAMDEG).

Assigning a value of PARAMDEG_DPSI that is greater than or equal to the number of qualified partitions for queries involving DPSIs allow DB2 to choose an access path for DPSIs that cut on partition boundaries. However, this decision depends on the available CPU resources (which potentially implies available zIIP resources). The parallel tasks triggered by DPSI parallelism are considered as CPU parallelism and eligible for zIIP processing. It is not expected that the value of PARAMDEG_DPSI would be lower than PARAMDEG. Also, the value of PARAMDEG_DPSI is not considered by the DB2 optimizer unless parallelism is enabled, and the query chooses an access path that uses a DPSI.

PARAMDEG_UTIL

PARAMDEG_UTIL specifies the maximum number of parallel suggests for the following utilities:

- REORG TABLESPACE
- REBUILD INDEX
- CHECK INDEX
- UNLOAD
- LOAD

Acceptable values are positive integers 0 - 32767, with a default of 0:

- 0: Utility determines the degree of parallelism and no additional constraint is placed on the maximum degree of parallelism in a utility.
- 1 to 32767: Specifies the maximum number of parallel suggests for all affected utilities.

The suggestion is to use the default value, but use the parameter to control the degree as needed to protect the overall system resources as virtual, real memory, and processor resource. The utility control statement PARALLEL overrides PARAMDEG_UTIL system parameter.

PCTFREE_UPD

The PCTFREE_UPD subsystem parameter specifies the default value to use for the PCTFREE FOR UPDATE clause of CREATE TABLESPACE or ALTER TABLESPACE statements. It specifies the default amount of free space to reserve on each page for use by subsequent UPDATE operations when data is added to the table by INSERT operations or utilities. This parameter has no effect on table spaces that have fixed-length rows.

Acceptable values are AUTO and 0 - 99, with a default of 0. This value is used as the default FOR UPDATE value when no PCTFREE FOR UPDATE clause is specified for a CREATE TABLESPACE or ALTER TABLESPACE statement:

- AUTO: DB2 uses real-time statistics values to automatically calculate the percentage of free space that is to be used by update operations. This value is equivalent to specifying PCTFREE FOR UPDATE -1 in the CREATE TABLESPACE or ALTER TABLESPACE statement.
- 0 - 99: DB2 reserves the specified percentage of space for use by update operations.

Without having experience with this function, you would assume that setting a value at page set level is more efficient in most cases than setting a positive integer for the system parameter. For details of PCTFREE_UPD, see 4.3, “Indirect references reduction” on page 88.

WFSTGUSE_AGENT_THRESHOLD

WFSTGUSE_AGENT_THRESHOLD determines the percentage of available space in the work file database on a DB2 subsystem or data sharing member that can be consumed by a single agent before a warning message is issued.

The percentage can range 0 - 100, with a default value of 0. A 0 means to disable this function. For a value greater than 0, DB2 issues a warning message when it detects the agent exceeds the work file usage threshold condition.

The suggestion is to turn on this function if there is a need to monitor the large usage of work file. Determine the action plan when the warning messages are issued. Refer to *DB2 11 for z/OS Technical Overview,* SG24-8180 for more details as the behavior is slightly different depending on the system parameter WFDBSEP (work file database separation between temporary work files and sort-related work files).

WFSTGUSE_SYSTEM_THRESHOLD

The WFSTGUSE_SYSTEM_THRESHOLD subsystem parameter determines the percentage of available space in the work file database on a DB2 subsystem or data sharing member that can be consumed by all agents before a warning message is issued.

Acceptable values are in the range 0 - 100. The default is set to 90%. When this value is 0, DB2 does not issue system-level space usage warnings for the work file database.

As a default behavior (90%), if WFDBSEP is NO, DB2 issues a warning message when the total system level storage used for declared global temporary tables (DGTTs) and work files together reaches or exceeds the 90% of the total configured storage in the WORKFILE database.

If WFDBSEP is YES, DB2 issues a warning message in the following two cases:

- When the total system level storage used for DGTTs reaches or exceeds the 90% of the total configured DGTT-storage in the WORKFILE database.
- When the total system-level storage used for work files reaches or exceeds the 90% of the total configured work files-storage in the WORKFILE database.

The suggestion is to keep the default and adjust the threshold value as needed.

11.6.2 Existing system parameters updated in DB2 11

Several system parameters have been updated on the upper limit or default value. Here are the affected system parameters that are related with performance tuning.

DSMAX

Upper limit increased from 100,000 to 200,000. The maximum number of data sets that can be open at one time.

See 3.5, “200,000 open data sets” on page 75 for details.

Important: If you observe a large number of data set close/open activities at the steady state or accounting shows the large wait time for open/close data sets, consider increasing DSMAX value. However, keep the value as low as you need since z/OS requires to use sizable 31-bit storage (4 KB per data set) to keep opening the data sets as demonstrated in 3.5, “200,000 open data sets” on page 75.

EDMDBDC

Defines the minimum size (in KB) of the DBD cache that is to be used by EDM at DB2 startup. Upper limit increased from 2097152 to 4194304. Default is 23400. The suggestion is to keep the value as you have at the migration, but to continue to monitor EDM statistics.

EDMSTMTC

Defines the size (in KB) of the statement cache that is to be used by the EDM. Upper limit increased from 1048576 to 4194304. Default is 113386. The suggestion is to keep the value as you have at the migration but continue to monitor global dynamic statement cache statistics.

EDM_SKELETON_POOL

Defines the minimum size of the EDM skeleton pool in KB that is to be used at DB2 start. Upper limit increased from 2097152 to 4194304. Default is 10240. The suggestion is to keep the value as you have at the migration but continue to monitor PT request and PT not found in EDM statistics.

MAXKEEPD

The total number of prepared, dynamic SQL statements that can be saved past a commit point by all threads in the system using KEEPDYNAMIC(YES) bind option. Upper limit increased from 65535 to 204800. Default is 5000. The suggestion is to keep the value as you have at the migration but continue to monitor local dynamic statement cache statistics.

REORG_LIST_PROCESSING

The default setting for the PARALLEL option of the DB2 REORG TABLESPACE utility. Default was changed from SERIAL to PARALLEL. The PARALLEL option indicates whether REORG TABLESPACE processes all partitions specified in the input LISTDEF statement in a single utility execution (PARALLEL YES) or process each in a separate utility execution (PARALLEL NO).

REORG_PART_SORT_NPSI

Specifies the default method of building a nonpartitioned secondary index (NPSI) during REORG TABLESPACE PART. Default was changed from NO to AUTO. This setting is used when the SORTNPSI keyword is not provided in the utility control statement. The SORTNPSI keyword specifies whether REORG TABLESPACE PART decides to sort all keys of an NPSI and how to make that decision. The setting is ignored for a REORG that is not part-level or a REORG with no NPSIs.

SUBQ_MIDX

Whether to enable or disable multiple index access for queries having subquery predicates. Default was changed from DISABLE to ENABLE.

12

Monitoring

DB2 11 continues to improve monitoring and tools support. Several DB2 instrumentation enhancements have been introduced. Some of the existing instrumentation facility component identifiers (IFCIDs) have been enhanced and some new IFCIDs were implemented. IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS (OMPE) V520 has been enhanced to support the DB2 11 IFCID enhancements.

DB2 11 has been enhanced to support some new tools also.

In this chapter, we describe the following topics:

- Instrumentation changes
- OMPE accounting batch report enhancements
- OMPE statistics batch report enhancements
- Reduction of Not Accounted time in the OMPE Accounting report
- Stored procedure monitoring OMPE report enhancements
- Hardware Instrumentation Services

12.1 Instrumentation changes

This section describes instrumentation enhancements in DB2 11 by additional IFCID fields and changes to existing IFCID fields.

12.1.1 Additional IFCID fields

Following are the IFCIDs introduced in DB2 11:

- IFCID 376: SQL statement compatibility issues
- IFCID 377: Records pseudo-deleted index entries are automatically cleaned up
- IFCID 27: Monitor sparse index usage
- IFCID 382 and 383: Records suspend operations for parallel task

A summary description is provided below. Refer to the DB2 provided PDS member *hlq*.SDSNIVPD(DSNWMSGS) for more detailed information about these IFCIDs.

> **Tip:** Refer to *Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS*, SG24-8182 for more details about loading IFCID field descriptions into a DB2 table for easier consultation.

IFCID 376: SQL statement compatibility issues

IFCID 376 is new and provides information about SQL statement compatibility issues with applications running in DB2 11 but in DB2 10 compatibility mode (BIND with APPLCOMPAT(V10R1)). This trace record is similar to the IFCID 0366 record, except that this trace record contains information for 0376 unique instances of SQL statements. The reason codes are interpreted in the section about IFCID 366 reason code (QW0366FN).

IFCID 377: Pseudo-deleted index entries are automatically cleaned up

DB2 11 adds automatic removal of pseudo-deleted index entries functionality. Refer to 4.2, "Pseudo-deleted index key cleanup" on page 83 for details about functionality and performance considerations.

IFCID 377 is introduced to monitor the index daemon activity when it cleans up committed pseudo-deleted entries from an index. It includes the DBID, PSID of the index being cleaned up, the partition number of the index, and the page number being cleaned up. It has an indicator to show if the cleanup is for a pseudo-empty page cleanup or for pseudo-deleted entry cleanup:

- For a pseudo-empty page cleanup, the pseudo-empty index page is deleted from the index tree.
- For a pseudo-deleted entry cleanup, the index page remains in the index tree and only the committed pseudo-deleted entries are removed from the index page.

It also has a field to show the number of pseudo-deleted entries removed from each index page. Example 12-1 shows the fields in IFCID 377.

Example 12-1 IFCID 377 fields

```
***********************************************************************
*   IFCID 0377 to record index pseudo delete daemon cleanup           *
***********************************************************************
*
QW0377   DSECT              IFCID(QWHS0377)
```

```
QW0377DB DS    CL2           DATA BASE ID
QW0377OB DS    CL2           INDEX PAGE SET ID
QW0377PT DS    CL2           PARTITION NUMBER
QW0377FL DS    CL1           FLAGS
         DS    CL1           RESERVED
QW0377PG DS    CL4           PAGE NUMBER of the index page cleaned up
QW0377NU DS    CL4           NUMBER OF PSEUDO DELETED ENTRIES REMOVED
QW0377DL EQU   X'80'         PSEUDO EMPTY PAGE IS DELETED
QW0377CL EQU   X'40'         PSEUDO DELETED ENTRIES CLEANED UP
*
```

The IFCID 377 record is written once per each index page being cleaned up. It is not included in any trace class since the trace volume can be large. To monitor the pseudo-deletion activity, start the performance trace for IFCID 377 for the wanted interval and format the trace data with OMPE job record trace option.

With the OMPE SYSIN syntax:

```
RECTRACE
INCLUDE (IFCID(377))
TRACE LEVEL(LONG)
```

You can get the IFCID 377 trace output. Example 12-2 shows a portion of an OMEGAMON XE for DB2 Performance Monitor on z/OS Record Trace Report with IFCID 377 details.

Example 12-2 OMPE Report IFCID 377

```
PRIMAUTH CONNECT   INSTANCE      END_USER      WS_NAME                       TRANSACT
ORIGAUTH CORRNAME  CONNTYPE      RECORD TIME   DESTNO ACE IFC  DESCRIPTION   DATA
PLANNAME CORRNMBR                TCB CPU TIME             ID
-------- -------- ----------- ----------------- ------ --- --- ------------- ------------------------------------------------------
SYSOPR   DB2B     CB6EA03AAC12 N/P             N/P                           N/P
SYSOPR   014.IDAE 'BLANK'     11:47:21.66633960    454   1 377 PSEUDO DELETE  NETWORKID:  DKBDON01  LUNAME:  BDPODTST  LUWSEQ:   144
'BLANK'  MK01                 N/P                             DAEMON CLEANUP
        |-------------------------------------------------------------------------------------------------------------------
        |DATABASE ID       :  403                      INDEX PAGE NUMBER  :         11       PARTITION NUMBER   :          1
        |INDEX PAGE SET ID  :  16                      PD ENTRIES REMOVED :        594
        |FLAG:             :  PAGE IS DELETED FROM INDEX
        |-------------------------------------------------------------------------------------------------------------------
                              11:47:21.67700906    455   1 377 PSEUDO DELETE  N/P
                              N/P                             DAEMON CLEANUP NETWORKID:  DKBDON01  LUNAME:  BDPODTST  LUWSEQ:   144
        |-------------------------------------------------------------------------------------------------------------------
        |DATABASE ID       :  403                      INDEX PAGE NUMBER  :         34       PARTITION NUMBER   :          1
        |INDEX PAGE SET ID  :  16                      PD ENTRIES REMOVED :        594
        |FLAG:             :  PAGE IS DELETED FROM INDEX
```

IFCID 27: Monitor sparse index usage

A DB2 sparse index is an array of join keys and record identifiers (RIDs). When no indexes are defined on the table and the table is small, a sparse index can greatly improve performance.

New IFCID 27 records what type of sparse index is used for probing, the amount of storage used, and many other characteristics about the current sparse index.

Also, additional fields are added to IFCID 2 and 3 to record if the sparse index could not be optimized because not enough storage was available, or if it had to be placed into a physical work file, which may hurt query performance.

With this new instrumentation, the user could be able to adjust their MXDTCACH[1] higher or lower depending on the storage available on their system to improve the query performance.

[1] System parameter for maximum storage allocated for data caching per thread.

The MXDTCACH subsystem parameter specifies the maximum amount of memory, in MB, that is to be allocated for data caching per thread. Acceptable values are 0 - 512; the default is 20.

IFCID 382 and 383: Monitor suspend operations for parallel task

Two new IFCIDs 382 and 383 are introduced in Accounting Classes 3 and 8 and Monitor Classes 3 and 8 to monitor the parallel task suspensions. They provide information about the begin and end of suspension for the parent and child parallel tasks. Example 12-3 lists the new IFCID 382 and 383 records fields.

Example 12-3 New IFCID 382 and 383 fields

```
***********************************************************************
* BEGIN Suspend for parallel task synchronization                     *
***********************************************************************
*QW0382       DSECT            IFCID(QWHS0382)
*QW0382ST     DS  CL1          Type of task suspended.
*QW0382PT     EQU C'P'         Task suspended is a parent
*QW0382CT     EQU C'C'         Task suspended is a child
*
***********************************************************************
* END Suspend for parallel task synchronization                       *
***********************************************************************
*QW0383       DSECT            IFCID(QWHS0383)
*QW0383RT     DS  CL1          Type of task resumed.
*QW0383PT     EQU C'P'         Task resumed is a parent
*QW0383CT     EQU C'C'         Task resumed is a child
*
```

12.1.2 Changes to existing IFCID fields

This section describes the DB2 11 enhancements to the existing IFCIDs:

IFCID 369: Aggregate accounting

IFCID 369 externalizes summary accounting values for each connect type at the statistics intervals. IFCID 369 was introduced in DB2 10.

IFCID 369 is added to STATISTICS CLASS 9 and externalized them every minute. IFCID 369 externalizes the summation of all agents that completed processing during this 1-minute interval.

IFCID 53 and 58 enhancements

IFCIDs 53 and 58 are the end SQL statement IFCIDs for a number of SQL statement types. To understand which SQL statement type a given IFCID 53/58 closes, it is necessary to correlate to a specific begin SQL statement (IFCID 66).

DB2 11 added a new statement identifier to IFCID 53 and IFCID 58. New fields QW0058TOS and QW0053TOS (type of SQL request) are introduced. IFCID 58 is identical to IFCID 53. If both IFCID 58 and IFCID 53 are started, those IFCID 58s without a begin SQL statement IFCID, are written as IFCID 53.

IFCID 2: Changes related to query parallelism monitoring

The following IFCID 2 fields were changed to track the effect to the degree of parallelism due to resource constraints. Table 12-1 on page 271 summarizes the changes.

Table 12-1 IFCID 2 changes related to parallelism

Name	Description
QXSTOREDGRP	Total number of parallel groups for which the degree of parallelism was reduced as a result of parallel system negotiation because system resources were constrained. Total number of parallel groups that changed to sequential mode as a result of parallel system negotiation because system resources were constrained.
QXSTODGNGRP	Number of parallel groups are degenerated to sequential due to system negotiation result of system stress level.
QXMAXESTIDG	The estimated maximum degree of parallelism for a parallel group. This value is estimated at bind time, based on the cost formula. If a parallel group contains a host variable or parameter marker, the estimate is based on assumed values.
QXMAXPLANDG	The planned maximum degree of parallelism for a parallel group. This value is the optimal degree of parallelism that can be obtained at execution time, after host variables or parameter markers are resolved, and before buffer pool negotiation and system negotiation are performed.
QXPAROPT	Total number of parallel groups that changed to sequential mode during optimization, for reasons such as the result of evaluation of a parallel group that zero rows are returned or a parallel group is partitioned on a single record.

IFCID 316: Changes related to query parallelism

DB2 11 adds fields to IFCID 316 that are related to query parallelism. Table 12-2 summarizes the changes.

Table 12-2 IFCID 316 changes related to parallelism

Name	Description
QW0316AVGESTI	Average estimated degree of parallelism for all parallel groups. The estimated degrees are calculated at bind time, based on the cost formula.
QW0316AVGPLAN	Average planned maximum degree of parallelism for all parallel groups. The average is calculated at execution time. It is the optimal degree of parallelism that can be obtained at execution time, after host variables or parameter markers are resolved, and before buffer pool negotiation and system negotiation are performed.
QW0316AVGACT	Average actual degree of parallelism for all parallel groups. The actual degree of parallelism is calculated at execution time, after buffer pool negotiation and parallel system negotiation are taken into account.

IFCID 3: QWACAWTW and QWACARNW fields

IFCID 3 QWACAWTW and QWACARNW fields were changed to include the Buffer Manager force write Class 3 suspension accounting time and events.

IFCID 3 and 239 enhancement to support autonomous transactions

IFCIDs 3 and 239 were enhanced to support autonomous transactions. Refer to the DSNDQWAS mapping macro.

IFCID 127 and 128 enhancement

IFCID 127 and 128 were enhanced to indicate Buffer Manager force type of write I/O wait.

IFCID changes for extended RBA and LRSN support

Numerous instrumentation changes were done to support a 10-byte relative byte address/log record sequence number (RBA/LRSN). Any application parsing the following records needs to be modified to fully use the new RBA size in these records. The following IFCIDs were changed:

- IFCID129
- IFCID306
- IFCID32
- IFCID34
- IFCID36
- IFCID38
- IFCID39
- IFCID43
- IFCID114
- IFCID119
- IFCID126
- IFCID129
- IFCID185
- IFCID203
- IFCID261
- IFCID306
- IFCID335

IFCID changes for temporal support

The following IFCIDs were changed to indicate the impacts of the CURRENT TEMPORAL BUSINESS_TIME special register, the CURRENT TEMPORAL SYSTEM_TIME special register, and the SYSIBMADM.GET_ARCHIVE built-in global variable in the field QWxxxxER (where xxxx is 0053, 0058, 0059,0060, 0061, 0064, 0065, 0066, and 0401.)

IFCID 225 change for Array support

IFCID 225 field QW0225AR was added to show total array variable storage.

IFCID 366 change for Application Incompatibility support

IFCID 366 (introduced in DB2 10) has been changed to identify applications that are affected by DB2 11 incompatible changes.

IFCID 230 and 256 change for Castout enhancement

IFCID 230 and 256 have been changed to include the new CLASST number of buffers value.

12.2 OMPE accounting batch report enhancements

This section describes the IBM Tivoli OMEGAMON XE for DB2 Performance Expert V520 (OMPE) batch report enhancements introduced to support DB2 11 instrumentation changes.

> **Tip:** Refer to *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference*, SH12-6991 for details about the topics described in this section.

OMPE accounting batch report has been enhanced to support the following new functions:

- Autonomous stored procedures
- Extended end user information (255 characters)
- Query parallelism improvements
- Array data type support
- Work file improvements
- Internal resource lock manager (IRLM) storage details enhancements
- GBP Sync Read(XI) hit ratio
- GBP Pages In Write-Around
- RID List improvements
- Statistics report CPU reporting enhancements

12.2.1 Autonomous stored procedures

Autonomous stored procedures are native procedures that execute under their own units of work, separate from the calling program. They commit when they finish without committing the work of the calling program. Because these procedures are independent of the calling program, the caller and the stored procedure each generate their own independent accounting records.

The OMPE batch accounting report has been enhanced to report the suspension time within the autonomous stored procedure. Example 12-4 shows an example of the OMPE accounting report sections that provide some details about the autonomous procedure.

Example 12-4 Accounting report sections for the autonomous procedure

```
HIGHLIGHTS
--------------------------
#OCCURRENCES    :   147644
#ALLIEDS        :        0
#ALLIEDS DISTRIB:        0
#DBATS          :   147644
#DBATS DISTRIB. :        0
#NO PROGRAM DATA:        9
#NORMAL TERMINAT:   147644
#DDFRRSAF ROLLUP:        0
#ABNORMAL TERMIN:        0
#CP/X PARALLEL. :    12920
#UTIL PARALLEL. :        0
#IO PARALLELISM :        0
#PCA RUP COUNT  :        0
#RUP AUTONOM. PR:        0  A
#AUTONOMOUS PR  :        0  B (QWAT_AT_COUNT)
#INCREMENT. BIND:        0
#COMMITS        :   147644
#ROLLBACKS      :        0
#SVPT REQUESTS  :        0
#SVPT RELEASE   :        0
#SVPT ROLLBACK  :        0
MAX SQL CASC LVL:        5
UPDATE/COMMIT   :     0.37
SYNCH I/O AVG.  : 0.002337
```

In Example 12-4 on page 273, legend **A** indicates the number of accounting records that indicated a roll-up autonomous thread and for legend **B**, the value depends on the record type:

- For non-rollup records, this value is the number of autonomous transactions executed.
- For parallel query rollup records, this value is 0.
- For DDF/RRSAF rollup records, this value is the number of autonomous transactions executed. These transactions are NOT counted in QWACPCNT. For rollup records, QWACPCNT is the aggregate transaction count.
- For autonomous transaction rollup records, this value is 0.

In Example 12-5, legend **C** shows the accumulated suspension time for the autonomous procedure.

Example 12-5 Class 3 suspension report

```
CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT
--------------------  ------------  --------
LOCK/LATCH(DB2+IRLM)      0.000618      1.14
 IRLM LOCK+LATCH          0.000112      0.07
 DB2 LATCH                0.000505      1.08
SYNCHRON. I/O             0.001793      0.77
 DATABASE I/O             0.001541      0.56
 LOG WRITE I/O            0.000252      0.21
OTHER READ I/O            0.000056      0.02
OTHER WRTE I/O            0.000000      0.00
SER.TASK SWTCH            0.000129      0.22
 UPDATE COMMIT            0.000000      0.00
 OPEN/CLOSE               0.000013      0.00
 SYSLGRNG REC             0.000002      0.00
 EXT/DEL/DEF              0.000011      0.00
 OTHER SERVICE            0.000103      0.22
ARC.LOG(QUIES)            0.000000      0.00
LOG READ                  0.000000      0.00
DRAIN LOCK                0.000001      0.00
CLAIM RELEASE             0.000000      0.00
PAGE LATCH                0.000008      0.02
NOTIFY MSGS               0.000000      0.00
GLOBAL CONTENTION         0.000791      2.64
COMMIT PH1 WRITE I/O      0.000000      0.00
ASYNCH CF REQUESTS        0.000041      0.33
TCP/IP LOB XML            0.000000      0.01
ACCELERATOR               0.000000      0.00
AUTONOMOUS PROCEDURE      0.000000      0.00 C (QWAC_AT_WAIT, QWAC_AT_COUNT)
PQ SYNCHRONIZATION        0.000581      0.45
TOTAL CLASS 3             0.004018      5.60
```

12.2.2 Extended end user information (255 characters)

The DB2 Client information fields are available on each distributed connection to a DB2 for z/OS database server. These fields are also available in other members of the DB2 family of databases. They enable a distributed application to provide additional information to DB2 that can be used to classify the workload within Workload Manager (WLM), or to filter accounting reports.

DB2 11 enhances Client information in DB2 for z/OS by expanding the lengths of these fields:

- Client User ID
- Client Application Name
- Client Workstation Name
- Client Accounting Information

Table 12-3 lists the Client information field length in DB2 11 for z/OS in contrast with the length of the same fields in DB2 10.

Table 12-3 Client information fields length changes with DB2 11 for z/OS

Field name	Maximum length in DB2 10	Maximum length in DB2 11
Client User ID	16 bytes	128 bytes
Client Application Name	32 bytes	255 bytes
Client Accounting Information	200 bytes	255 bytes
Client Workstation Name	18 bytes	255 bytes
Client Correlation Token	N/A	255 bytes

The longer client information length provides better granularity in the exploitation of this information, and compatibility with other database members of the DB2 family of products. The longer client information strings are used in:

- WLM enclave classification
- DB2 supplied SYSPROC.WLM_SET_CLIENT_INFO stored procedure
- RRSAF DSNRLI SET_CLIENT_ID function
- Rollup accounting
- Profile Monitoring for remote threads and connections
- Resource limit facility (RLF)
- Client information special registers
- DISPLAY THREAD command output
- Various trace records, like IFCID 172, 196, 313, and 316

Example 12-6 is an OMPE accounting trace long report example showing the extended end user name, transaction name and the work station name. In the Accounting identification block, the long names might be truncated. The full name is shown at the end of the report.

Example 12-6 Accounting - Truncated values

```
---- IDENTIFICATION --------------------------------------------------------------------------------------------------------------
 ACCT TSTAMP: 13-09-03 21:25:47.05    PLANNAME: MyAppNam       WLM SCL: DDFBAT          CICS NET: N/A
 BEGIN TIME : 13-09-03 21:25:30.65    PROD TYP: JDBC DRIVER                             CICS LUN: N/A
 END TIME   : 13-09-03 21:25:47.05    PROD VER: V4 R16M0       LUW NET: G9378992        CICS INS: N/A
 REQUESTER  : ::9.55.137.146          CORRNAME: db2jcc_a       LUW LUN: D447
 MAINPACK   : SYSLH200                CORRNMBR: ppli           LUW INS: CBE919CC84BB    ENDUSER : YasuhiroOhmori#1
 PRIMAUTH   : DB2R6                   CONNTYPE: DRDA           LUW SEQ:            2    TRANSACT: MyAppName012345678901234567890#1
 ORIGAUTH   : DB2R6                   CONNECT : SERVER                                  WSNAME  : MyClientComputer#2

      -------------------------------------------------------------------------------------------------------------------------------
      |TRUNCATED VALUE                  FULL VALUE                                                                                    |
      |MyAppName012345678901234567890#1 MyAppName012345678901234567890123456789012345678901234567890123456789012345678901234567890123|
      |                                 456789A123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456|
      |                                 7890123456789B1234567890123456789012345678901234567890123456789012345                         |
      |MyClientComputer#2               MyClientComputer7890123456789012345678901234567890123456789012345678901234567890123456789 0123|
      |                                 456789A123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890123456|
      |                                 7890123456789B1234567890123456789012345678901234567890123456789012345                         |
      |YasuhiroOhmori#1                 YasuhiroOhmori5678901234567890123456789012345678901234567890123456789012345678901234567890123|
      |                                 456789A1234567890123456789012345678                                                           |
      -------------------------------------------------------------------------------------------------------------------------------
```

12.2.3 Query parallelism

If a query uses query central processor (CP) parallelism, several tasks (called *parallel tasks*) perform the work. For each of these tasks an accounting record is generated, which contains counters and timers pertinent to the work performed by the particular task. In addition, an accounting record is created that contains the details about nonparallel work within the thread and data related to parallel work.

Example 12-7 shows the layout for the OMPE accounting report query parallelism section and the new OMPE fields have been highlighted.

Example 12-7 Accounting and statistics report query parallelism section

```
QUERY,PARALLELISM,              AVERAGE     TOTAL
----------------------------  --------  --------
MAXIMUM DEGREE-ESTIMATED          0.00        12 (QXMAXESTIDG)
MAXIMUM DEGREE-PLANNED            0.00        12 (QXMAXPLANDG)
MAXIMUM DEGREE-EXECUTED            N/A        12 (QXMAXDEG)
MAXIMUM MEMBERS USED               N/A         0
GROUPS EXECUTED                   0.12     17324
 RAN AS PLANNED                   0.12     17322
 RAN REDUCED-STORAGE              0.00         0 (QXREDGRP)
 RAN REDUCED-NEGOTIATION          0.00         0 (QXSTOREDGRP)
 SEQ-CURSOR                       0.00         0 (QXDEGCUR)
 SEQ-NO ESA SORT                  0.00         0 (QXDEGESA)
 SEQ-NO BUFFER                    0.00         0 (QXDEGBUF)
 SEQ-ENCLAVE SERVICES              N/A       N/A (QXDEGENC)
 SEQ-AUTONOMOUS PROCEDURE         0.00         0 (QXDEGAT)
 SEQ-NEGOTIATION                  0.00         0 (QXSTODGNGRP)
 ONE DB2-COORDINATOR = NO         0.00         0
 ONE DB2-ISOLATION LEVEL          0.00         0
 ONE DB2-DCL TEMPORARY TABLE      0.00         0
MEMBER SKIPPED (%)                 N/C       N/A
DISABLED BY RLF                   0.00         0
REFORM PARAL-CONFIG               0.01      1101
REFORM PARAL-NO BUF               0.00         0
```

See *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991 for the details about the fields.

12.2.4 Array data type support

DB2 11 provides support for the array data type. An array data type is a user-defined data type that is an ordinary array or an associative array. The elements of an array type are based on one of the existing built-in data types. Currently, array data type is not supported in other contexts such as columns of tables and views, triggers, and client interfaces that are not essential for migrating applications.

The CREATE TYPE (array) statement defines an array data type. The SYSIBM.SYSDATATYPES table contains one row for each array data type defined.

The array data type can only be used as a data type of:

- An SQL variable
- A parameter or RETURNS data type of an SQL scalar function

- A parameter of a native SQL procedure
- The target data type for a CAST specification

Tip: Refer to the IBM *DB2 11 for z/OS Technical Overview*, SG24-8180 for more details about DB2 11 array type support.

12.2.5 Sparse index enhancement

In Version 8 of DB2 for z/OS, materialized work files for star join queries can be stored in the global storage pool above the 2 GB bar (an in-memory data cache) as a more cost-effective alternative to the sparse index technique that is used in prior versions of DB2 for z/OS. A sparse index is used as a fallback at run time if enough memory is not available in the global storage pool.

In DB2 9, the use of in-memory data caching was generalized to support more types of queries. In addition, data is cached in a local storage pool above the 2 GB bar, which can reduce potential storage contention because data caching storage management is associated with each thread.

In DB2 10, you can allocate up to 20 MB of memory per thread for data caching. If the required memory exceeds the specified limit, a sparse index is used instead. This expanded access method provides potential benefits for tables that lack an appropriate index or enough statistics, and competes with other options for the most efficient access path.

DB2 11 introduces IFCID 27 to monitor sparse index usage. OMPE reports the new fields related to Array data and Sparse index in the accounting miscellaneous section shown in Example 12-8.

Example 12-8 Accounting miscellaneous section

```
MISCELLANEOUS,          AVERAGE     TOTAL
--------------------   --------   --------
MAX STO LOB VAL (KB)       0.02          2
MAX STO XML VAL (KB)      10.51        788
ARRAY EXPANSIONS           0.00          0 D (QXSTARRAY_EXPANSIONS)
SPARSE IX DISABLED         0.00          0 E (QXSISTOR)
SPARSE IX BUILT WF         0.00          0 F (QXSIWF)
```

In Example 12-8:

- Legend **D** indicates the number of times a variable array was expanded beyond 32 KB.
- Legend **E** indicates the number of times the sparse index was disabled because of insufficient storage.
- Legend **F** indicates the number of times the sparse index processing built a work file for probing.

12.3 OMPE statistics batch report enhancements

OMPE statistics batch report has been enhanced to support the following new functions:

- Support work file improvements
- IRLM storage details
- GBP Sync Read(XI) hit ratio

- GBP Pages In Write-Around
- RID list improvements

12.3.1 Support work file improvements

Example 12-9 shows the OMPE Statistics report long layout for work file usage. The newly reported fields are highlighted.

Example 12-9 Statistics report long

```
WORKFILE DATABASE           QUANTITY  /SECOND  /THREAD  /COMMIT
--------------------------  --------  -------  -------  -------
TOTAL STORAGE CONFIG   (KB)   256.00      N/A      N/A      N/A (QISTWSTG)
  TOT DGTT STOR CONFIG (KB)   128.00      N/A      N/A      N/A (QISTDGTTSTG)
  TOT WF STOR CONFIG   (KB)   128.00      N/A      N/A      N/A (QISTWFSTG)
TOTAL STORAGE THRESHOLD (%)    90.00      N/A      N/A      N/A (QISTSSTH)

MAX TOTAL STORAGE USED (KB)   128.00      N/A      N/A      N/A (QISTWMXU)
  MAX DGTT STOR USED   (KB)    64.00      N/A      N/A      N/A (QISTDGTTMXU)
  MAX WF STOR USED     (KB)    64.00      N/A      N/A      N/A (QISTWFMXU)

CUR TOTAL STORAGE USED (KB)     0.00      N/A      N/A      N/A (QISTWCTO)
  CUR DGTT STOR USED   (KB)     0.00      N/A      N/A      N/A (QISTDGTTCTO)
  CUR WF STOR USED     (KB)     0.00      N/A      N/A      N/A (QISTFCTO)
  STORAGE IN 4K TS     (KB)     0.00      N/A      N/A      N/A
  STORAGE IN 32K TS    (KB)     0.00      N/A      N/A      N/A

4K USED INSTEAD OF 32K TS       0.00     0.00      N/C     0.00
32K USED INSTEAD OF 4K TS       0.00     0.00      N/C     0.00

MAX ACTIVE (DM) IN-MEMORY      34.00      N/A      N/A      N/A
  MAX ACT (NONSORT) IN-MEM      0.00      N/A      N/A      N/A (QISTI2AH)
CUR ACTIVE (DM) IN-MEMORY       0.50      N/A      N/A      N/A
  CUR ACT (NONSORT) IN-MEM      0.00      N/A      N/A      N/A (QISTI2AC)
MAX STOR (DM) IN-MEM   (KB)  1088.00      N/A      N/A      N/A
CUR STOR (DM) IN-MEM   (KB)    16.00      N/A      N/A      N/A
MAX ACTIVE (SORT) IN-MEMORY     0.00      N/A      N/A      N/A
CUR ACTIVE (SORT) IN-MEMORY     0.00      N/A      N/A      N/A
MAX STOR (SORT) IN-MEM (KB)     0.00      N/A      N/A      N/A
CUR STOR (SORT) IN-MEM (KB)     0.00      N/A      N/A      N/A
IN-MEM (NONSORT) OVERFLOWED     0.00     0.00      N/C     0.00 (QISTI2OF)
IN-MEM WORKF NOT CREATED        0.00     0.00      N/C     0.00 (QISTIMNC)
AGENT STORAGE CONFIG   (KB)     0.00      N/A      N/A      N/A (QISTWMXA)
  NUMBER OF LIMIT EXCEEDED      0.00     0.00      N/C     0.00 (QISTWFNE)
AGENT STORAGE THRESHOLD (%)    90.00      N/A      N/A      N/A (QISTASTH)
MAX AGENT STORAGE USED (KB)    64.00      N/A      N/A      N/A (QISTAMXU)
```

See *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991 for the details about the fields.

12.3.2 IRLM storage details

Example 12-10 on page 279 shows the IRLM storage details of a more informative OMPE statistics report.

Example 12-10 IRLM storage details

```
IRLM STORAGE BELOW AND ABOVE 2 GB                          QUANTITY
--------------------------------------------  ------------------
EXTENDED CSA SIZE IN USE                (MB)                1.20
HWM EXTENDED CSA SIZE IN USE            (MB)                1.20

31 BIT PRIVATE IN USE                   (MB)                5.41
HWM 31 BIT PRIVATE IN USE               (MB)                5.41
THRESHOLD 31 BIT PRIVATE                (MB)                0.00

64 BIT PRIVATE IN USE                   (MB)                0.00
HWM 64 BIT PRIVATE IN USE               (MB)                0.00
THRESHOLD 64 BIT PRIVATE                (MB)             1494.00

64 BIT COMMON IN USE                    (MB)                0.00
HWM 64 BIT COMMON IN USE                (MB)                0.00
```

See *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991 for the details about the fields.

12.3.3 GBP Sync Read(XI) hit ratio

OMPE Statistics report long now shows the GBP Sync Read(XI) hit ratio to help to tune GBP size, as shown in Example 12-11.

Example 12-11 Statistics report long

```
GROUP BP10                     QUANTITY  /SECOND  /THREAD  /COMMIT
----------------------------  --------  -------  -------  -------
GROUP BP R/W RATIO (%)           67.56      N/A      N/A      N/A
GBP SYN.READ(XI) HIT RATIO(%)   100.00      N/A      N/A      N/A
GBP-DEPENDENT GETPAGES        16764.00     4.74     0.34     0.34
SYN.READ(XI)-DATA RETURNED     3535.00     1.00     0.07     0.07
SYN.READ(XI)-NO DATA RETURN       0.00     0.00     0.00     0.00
SYN.READ(NF)-DATA RETURNED     2862.00     0.81     0.06     0.06
SYN.READ(NF)-NO DATA RETURN       1.00     0.00     0.00     0.00
UNREGISTER PAGE                1875.00     0.53     0.04     0.04
...........
```

If the GBP Sync Read(XI) hit ratio is below 90% during critical performance periods, the GBP size needs to be increased. The GBP Sync Read(XI) hit ratio is calculated as:

```
SYN.READ(XI)-DATA RETURNED / (SYN.READ(XI)-DATA RETURNED + SYN.READ(XI)-NO DATA
RETURN) X 100
(QBGLXD / (QBGLXR + QBGLXD))
```

12.3.4 GBP Pages In Write-Around

OMPE Statistics report long now shows the GBP Pages In Write-Around as shown in Example 12-12 on page 280. GBP write-around occurs when the updated pages are written to DASD when certain thresholds leading to GBP full condition are detected.

Example 12-12 Statistics report long: Additional GBP Pages In Write-Around

```
GROUP BP10   CONTINUED   QUANTITY  /SECOND  /THREAD  /COMMIT
-----------------------  --------  -------  -------  -------
WRITE AND REGISTER        7046.00     1.99     0.12     0.12
WRITE AND REGISTER MULT      7.00     0.00     0.00     0.00
CHANGED PGS SYNC.WRTN     7062.00     2.00     0.12     0.12
CHANGED PGS ASYNC.WRTN       0.00     0.00     0.00     0.00
PAGES WRITE & REG MULT      16.00     0.00     0.00     0.00
READ FOR CASTOUT             0.00     0.00     0.00     0.00
READ FOR CASTOUT MULT        0.00     0.00     0.00     0.00

PAGE P-LOCK LOCK REQ     10659.00     3.01     0.19     0.19
 SPACE MAP PAGES             0.00     0.00     0.00     0.00
 DATA PAGES                  0.00     0.00     0.00     0.00
 INDEX LEAF PAGES        10659.00     3.01     0.19     0.19

PAGE P-LOCK UNLOCK REQ   10656.00     3.01     0.19     0.19
PAGE P-LOCK LOCK SUSP       12.00     0.00     0.00     0.00
 SPACE MAP PAGES             0.00     0.00     0.00     0.00
 DATA PAGES                  0.00     0.00     0.00     0.00
 INDEX LEAF PAGES           12.00     0.00     0.00     0.00

PAGE P-LOCK LOCK NEG         3.00     0.00     0.00     0.00
 SPACE MAP PAGES             0.00     0.00     0.00     0.00
 DATA PAGES                  0.00     0.00     0.00     0.00
 INDEX LEAF PAGES            3.00     0.00     0.00     0.00

 PAGES IN WRITE-AROUND        0.00     0.00     0.00     0.00 (QBGLWA)
```

12.3.5 RID list improvements

The OMPE Statistics report long has been improved to show additional RID list metrics as shown in Example 12-13.

Example 12-13 Statistics report long additional RID list

```
RID LIST PROCESSING          QUANTITY  /SECOND  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------
SUCCESSFUL                    8528.1K    14.2K      N/C     8.41 (QXMIAP)
NOT USED-NO STORAGE              0.00     0.00      N/C     0.00 (QXNSMIAP)
NOT USED-MAX LIMIT               0.00     0.00      N/C     0.00 (QXMRMIAP)

MAX RID BLOCKS ALLOCATED        35.00      N/A      N/A      N/A
CURRENT RID BLOCKS ALLOCAT.      0.40      N/A      N/A      N/A

MAX RID BLOCKS OVERFLOWED        0.00      N/A      N/A      N/A
CURRENT RID BLOCKS OVERFL.       0.00      N/A      N/A      N/A

TERMINATED-NO STORAGE            0.00     0.00      N/C     0.00
TERMINATED-EXCEED RDS LIMIT      0.00     0.00      N/C     0.00
TERMINATED-EXCEED DM LIMIT       0.00     0.00      N/C     0.00
TERMINATED-EXCEED PROC.LIM.      0.00     0.00      N/C     0.00
OVERFLOWED-NO STORAGE            0.00     0.00      N/C     0.00 (QXWFRIDS)
OVERFLOWED-MAX LIMIT             0.00     0.00      N/C     0.00 (QXWFRIDT)
```

```
INTERRUPTED-NO STORAGE          0.00     0.00      N/C      0.00 (QXHJINCS)
INTERRUPTED-MAX LIMIT           0.00     0.00      N/C      0.00 (QXHJINCT)
SKIPPED-INDEX KNOWN             0.00     0.00      N/C      0.00 (QXRSMIAP)
```

See *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991 for the details about the fields.

Example 12-14 shows the list of the descriptions of the RID fields.

Example 12-14 RID list: Additional fields

```
QXMIAP                      # of times RID list processing used. This
*                           field is incremented once for a given
*                           table access for Index Access with list
*                           prefetch and for Multiple Index Access.
QXNSMIAP                    # of times a RID list was not used for
*                           a given RID list process involving one
*                           index (Index Access with list prefetch)
*                           or involving multiple indexes (Multiple
*                           Index Access) because no storage was
*                           available to hold the list of RIDs.
QXMRMIAP                    # of times one or more RID lists were not
*                           used for a given RID list process involving
*                           one index (Index Access with list prefetch)
*                           or involving multiple indexes (Multiple
*                           Index Access) because the number of RIDs
*                           exceeded one or more internal limits.
QXWFRIDS                    # of times a RID list was overflown to a
*                           work file because no RIDPOOL storage was
*                           available to hold the list of RIDs.
QXWFRIDT                    # of times a RID list was overflown to a
*                           work file because the number of RIDs
*                           exceeded one or more internal limits.
QXHJINCS                    # of times a RID list append for a Hybrid
*                           Join was interrupted because no RIDPOOL
*                           storage was available to hold the list of
*                           RIDs (i.e. number of times DB2
*                           interrupted the RID phase and switched to
*                           the Data phase).
QXHJINCT                    # of times a RID list append for a Hybrid
*                           Join was interrupted because the number
*                           of RIDs exceeded one or more internal
*                           limits (i.e. number of times DB2
*                           interrupted the RID phase and switched to
*                           the Data phase).
QXRSMIAP                    # of times a RID list retrieval for
*                           multiple index access was skipped because
*                           it was not necessary due to DB2 being
*                           it was not necessary due to DB2 being
*                           able to predetermine the outcome of index
*                           ANDing or ORing.
```

12.3.6 Statistics report CPU reporting enhancements

Following are some OMPE Statistics report enhancements that were introduced with DB2 10 after the general availability:

- In the Statistics address space CPU section, the last column heading now correctly shows 'CP CPU/Commit' as it does not include CPU consumed on zIIP. Previously the column heading was 'Total CPU/Commit'.
- DB2 now captures aggregated accounting data for the statistics interval for the different connect type in IFCID 369. IFCID 369 is started using the **START TRACE** command for STATISTICS CLASS(9).

IFCID 3 must also be enabled (ACCOUNTING CLASS(1)) to get IFCID 369 trace records.

IFCID 369 values are aggregated each time an IFCID 3 is written and it contains total values. In the Statistics reports, IFCID 369 delta values are calculated to show which IFCID 3 events occurred in a Statistics interval.

In contrast to Accounting reports, the IFCID 369 statistics performance metrics cannot distinguish between IFCID 3 records for parallel and non-parallel threads.

In Accounting, the elapsed times of parallel threads are derived from IFCID 3 of the originating thread. In IFCID 369, all times (even the elapsed times of the originating thread and parallel subtasks) are aggregated and shown as such in the Statistics report. That is why IFCID 369 can provide more diagnostics on what is happening in a DB2 subsystem and whether there are bottlenecks (also for parallel subtasks).

Example 12-15 shows a sample OMPE statistics report for the two enhancements with the changes in bold and green.

Example 12-15 Enhanced Statistics report

```
CPU TIMES                              TCB TIME     PREEMPT SRB  NONPREEMPT SRB     CP CPU TIME  PREEMPT IIP SRB  CP CPU /COMMIT
-------------------------------  ---------------  ---------------  ---------------  ---------------  ---------------  --------------
SYSTEM SERVICES ADDRESS SPACE          19.529808        0.250513        3.999053       23.779374        17.988831        0.006812
DATABASE SERVICES ADDRESS SPACE         1.508536        0.149497        0.039359        1.697391         6.389113        0.000486
IRLM                                    0.000066        0.000000        2.256474        2.256540         0.000000        0.000646
DDF ADDRESS SPACE                       0.141870        1.262984        0.022711        1.427565         0.701908        0.000409

TOTAL                                  21.180280        1.662993        6.317596       29.160870        25.079851        0.008353

CONNTYPE   CL1 ELAPSED      CL1 CPU   CL1 SE CPU  CL2 ELAPSED      CL2 CPU   CL2 SE CPU    CL3 SUSP  CL2 NOT ACC  QUANTITY
--------  ------------ ------------ ------------ ------------ ------------ ------------ ------------ ------------  --------
BATCH      5:45.700412    17.102508     0.000000  4:32.481540    14.523955     0.000000    18.522110  3:59.443103    126.00
CICS       3:35.843506     2.622503     0.000000  1:20.432247     2.010811     0.000000    17.077670  1:01.343766    144.00
DDF           2.967722     0.431580     0.606128     2.042811     0.380759     0.526230     0.438922     1.223130   2826.00
IMS                N/P          N/P          N/P          N/P          N/P          N/P          N/P          N/P      0.00
RRSAF     1:01:02.9662    42.948526     0.000000 16:07.659323    39.667408     0.000000  3:09.608421 12:20.389787    282.00
UTILITY    5:09.584462     6.927362     5.508104  2:23.883903     2.295560     5.508104    22.666686  1:58.921657     11.00,
```

Figure 12-1 Enhanced Statistics report

12.4 Reduction of Not Accounted time in the OMPE Accounting report

This section describes improvements done in DB2 11 to reduce the reported Not Accounted time in the DB2 accounting class 3 suspension trace and reports:

- Format write I/O suspension time
- Parallel query parent-child synchronization time
- Accounting Class 3 Suspensions report

12.4.1 Format write I/O suspension time

DB2 Buffer Manager format write I/O suspension time is now captured in the DB2 accounting Class 3 “Other Write I/O” suspension time.

When SQL processing appends data to the end of a DB2 data set, DB2 asynchronously preformats the additional space so that parallel update writes can be used. Latching delays caused by preformatting is captured as class 3 latch suspension time, and delays due to preformat the first 16 cylinders of a new extent are captured as class 3 DEF/EXT/DEL suspension time.

However, SHRLEVEL NONE utilities do not work the same way. Neither does the shadow data set that is written by online REORG. Because there is no need to share the data set, it can be formatted along with the data using “format writes”. When a utility does a Commit or when the horizontal deferred write threshold is reached, the utility must be suspended while the buffers are forced to DASD. Before DB2 11, this suspension time was not captured. DB2 11 captures this time as “Other Write I/O” suspension.

12.4.2 Parallel query parent-child synchronization time

Parallel query (PQ) parent task and child tasks synchronization time is now captured in DB2 accounting Class 3 suspension counters, as shown in Example 12-16 and Example 12-17.

Example 12-16 Parallel query (PQ) parent task and child tasks synchronization time

```
PQ Synchronization wait time :  QWAC_PQS_WAIT
PQ Synchronization events : QWAC_PQS_COUNT
```

12.4.3 Accounting Class 3 Suspensions report

Example 12-17 shows the OMPE accounting report that shows the PQ Synchronization information.

Example 12-17 OMPE Accounting Class 3 Suspensions

```
CLASS 3 SUSPENSIONS   AVERAGE TIME  AV.EVENT
--------------------  ------------  --------
LOCK/LATCH(DB2+IRLM)      0.000618      1.14
 IRLM LOCK+LATCH          0.000112      0.07
 DB2 LATCH                0.000505      1.08
SYNCHRON. I/O             0.001793      0.77
 DATABASE I/O             0.001541      0.56
 LOG WRITE I/O            0.000252      0.21
OTHER READ I/O            0.000056      0.02
OTHER WRTE I/O            0.000000      0.00 QWACAWTW (Format Write I/O suspension)
SER.TASK SWTCH            0.000129      0.22
 UPDATE COMMIT            0.000000      0.00
 OPEN/CLOSE               0.000013      0.00
 SYSLGRNG REC             0.000002      0.00
 EXT/DEL/DEF              0.000011      0.00
 OTHER SERVICE            0.000103      0.22
ARC.LOG(QUIES)            0.000000      0.00
LOG READ                  0.000000      0.00
DRAIN LOCK                0.000001      0.00
CLAIM RELEASE             0.000000      0.00
PAGE LATCH                0.000008      0.02
NOTIFY MSGS               0.000000      0.00
```

```
GLOBAL CONTENTION         0.000791      2.64
COMMIT PH1 WRITE I/O      0.000000      0.00
ASYNCH CF REQUESTS        0.000041      0.33
TCP/IP LOB XML            0.000000      0.01
ACCELERATOR               0.000000      0.00
AUTONOMOUS PROCEDURE      0.000000      0.00
PQ SYNCHRONIZATION        0.000581      0.45 QWAC_PQS_WAIT (Parallel Query synch)
TOTAL CLASS 3             0.004018      5.60
```

12.5 Stored procedure monitoring OMPE report enhancements

This section describes the stored procedure and user-defined function (UDF) instrumentation IFCID enhancements that were introduced in DB2 10 after general availability with APAR PM53243:

- Stored procedure monitoring overview
- Changed and new IFCIDs for stored procedure and UDF monitoring

12.5.1 Stored procedure monitoring overview

Stored procedure and UDF performance and tuning analysis has typically been performed via a combination of IFCID 3 and IFCID 239:

- IFCID 3 provides plan level information and aggregates all executions of stored procedures or UDFs into common fields. This can create difficulty when tuning multiple procedures or functions that are executed in a given transaction.
- IFCID 239 is also used for performance and tuning analysis at the package level. This provides better granularity than IFCID 3 but still might not be sufficient for all transactions because multiple package executions are reported together and CPU, elapsed, and suspend time reflect averages across many stored procedure packages.

If a procedure or function is executed multiple times, the variation between executions cannot be identified. Instrumentation enhancements were needed. DB2 implements multiple IFCID enhancements to provide more effective performance and tuning analysis of stored procedures and user-defined functions.

12.5.2 Changed and new IFCIDs for stored procedure and UDF monitoring

Following are the changes in the instrumentation facility in DB2 11:

- IFCID 233 is written at the beginning and end of a stored procedure or user-defined function invocation. This record is enhanced with the invoking statement ID, the invoking statement type, the version ID (applies only to versioned procedures), and the routine ID. The routine ID may be zero if a REBIND is not performed for packages containing CALL statements where the stored procedure name is a literal.
- New IFCIDs 380 and 381 are created for stored procedure (IFCID 380) and user-defined function (IFCID 381) details. These records have two data sections. Data section 1 is mapped by QW0233. Data section 2 is mapped by QW0380, which includes CP, specialty engine, and elapsed time details for nested activity. A series of IFCIDs 380 and 381 records can be used to determine the amount of class 1 and class 2 CP, specialty engine, and elapsed time relative to the execution of a given stored procedure or user-defined function.

- New IFCIDs 497, 498, and 499 are created for statement level detail. These records track dynamic and static DML statements executed by a transaction, including those executed within a stored procedure or user-defined function.
- New IFCID 497 is the statement ID detail record for statements executed outside of a stored procedure or UDF environment.
- IFCID 498 is the statement ID detail record for statements executed inside of a UDF environment. This includes WLM UDFs and non-inline scalar functions.
- IFCID 499 is the statement ID detail record for statements executed inside of a stored procedure environment. This includes both WLM and native stored procedures.

Any packages containing static SQL statements that existed before DB2 10 must be rebound in DB2 10 NFM to obtain a valid statement ID.

A new performance class 24 has been created to encapsulate IFCID 380 and IFCID 499 for stored procedure detail analysis.

For users who are interested in using the functions provided, the following actions need to be considered:

- For a CALL statement to a DB2 for z/OS stored procedure, the stored procedure name can be identified via a literal or by using a host variable or parameter marker. When using a literal for the stored procedure name, and in order to benefit from the enhancement that provides a valid routine ID in various IFCID records, the packages that contain the CALL statement must be rebound.
- For an SQL statement that invokes a DB2 for z/OS user-defined function in order to benefit from the enhancement to provide a valid routine ID in various IFCID records, the packages that contain the SQL statement must be rebound.
- The mapping of IFCID 233 remains compatible with prior versions and no immediate change is required. However, applications that parse this record need to be changed in order to use the new fields.

Figure 12-2 on page 286 summarizes the IFCID records written for stored procedure monitoring.

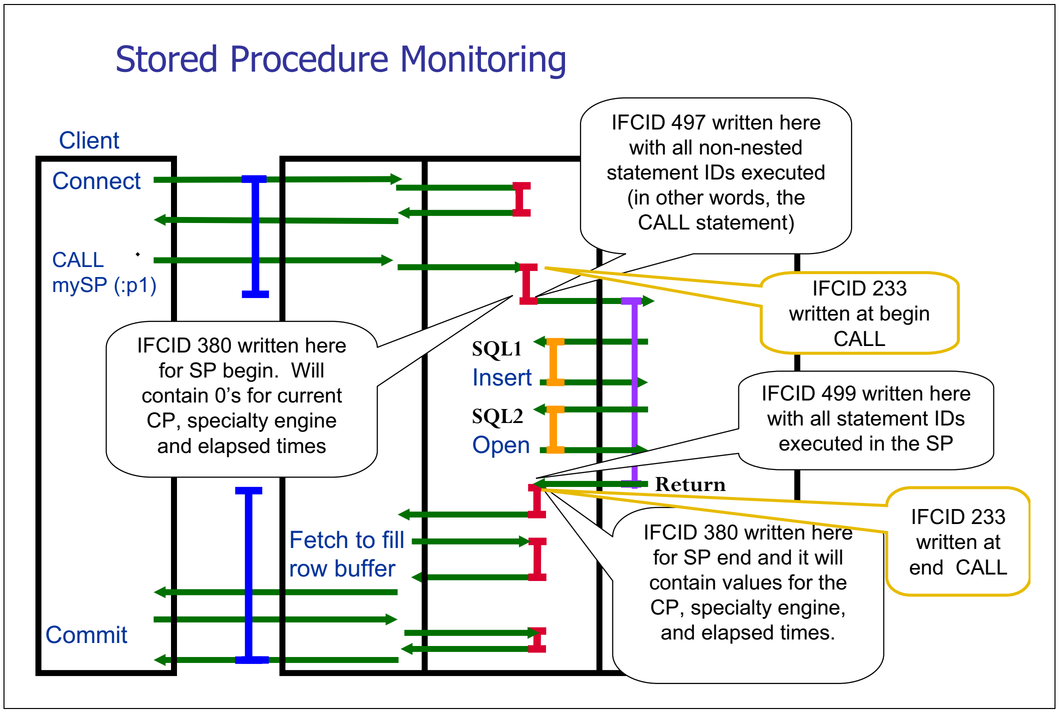

Figure 12-2 Monitoring stored procedures

If a trace was active to collect IFCIDs 233, 380, 497, and 499, following are the records that would be written:

- 497 to record the beginning of a call statement with the statement ID that triggered the call.
- 233 to record the beginning of the call statement. It shows the calling program's name, schema collection ID, and so on, as well as the routine ID and the nesting level.
- 380 to provide the beginning stored procedure detail record with beginning times.
- 499 to record the end of a call statement along with the statement IDs that the stored procedure executed and how many times they were executed.
- 233 to show the end of the stored procedure with the SQL code and the SQLCA of the call statement.
- 380 to show the ending class 1 and class 2 times for the stored procedure.

For more information, see the descriptions of the IFCIDs in *hlq*.SDSNIVPD(DSNWMSGS) and macros in *hlq*.SDSNMACS.

OMPE batch accounting report has been enhanced to order the report by activity name (ACTNAME). The accounting report reports the packages by the activity name such as stored procedure or UDF or trigger.

Example 12-18 shows the OMPE accounting report with ORDER(ACTNAME).

Example 12-18 OMPE accounting report with ORDER(ACTNAME)

```
LOCATION: D2ATOI01                 OMEGAMON XE FOR DB2 PERFORMANCE EXPERT (V5R2M0)                 PAGE: 1-1
      GROUP: N/P                             ACCOUNTING REPORT - LONG                     REQUESTED FROM: ALL       10:00:00.00
     MEMBER: N/P                                                                                     TO: DATES     17:01:00.00
  SUBSYSTEM: DATO                                  ORDER: ACTNAME                          INTERVAL FROM: 06/15/13 10:23:34.31
DB2 VERSION: V10                                  SCOPE: MEMBER                                      TO: 06/15/13 16:46:29.58

ACTNAME: NATIVE SQL PROC.SPDBA001
```

```
SPDBA001             VALUE                SPDBA001                  TIMES    SPDBA001             AVERAGE TIME  AVG.EV    TIME/EVENT
------------------   ------------------   ------------------  ------------    ------------------   ------------  ------  ------------
TYPE                 PACKAGE              ELAP-CL7 TIME-AVG       0.011228    LOCK/LATCH               0.000000    0.00           N/C
                                          CP CPU TIME             0.008470     IRLM LOCK+LATCH         0.000000    0.00           N/C
LOCATION             'BLANK'                AGENT                 0.008470     DB2 LATCH               0.000000    0.00           N/C
COLLECTION ID        'BLANK'                PAR.TASKS             0.000000    SYNCHRONOUS I/O          0.000000    0.00           N/C
PROGRAM NAME         'BLANK'              SE CPU TIME             0.014449    OTHER READ I/O           0.000000    0.00           N/C
                                          SUSPENSION-CL8          0.000000    OTHER WRITE I/O          0.000000    0.00           N/C
ACTIVITY TYPE        NATIVE SQL PROC        AGENT                 0.000000    SERV.TASK SWITCH         0.000000    0.00           N/C
ACTIVITY NAME        SPDBA001               PAR.TASKS             0.000000    ARCH.LOG(QUIESCE)        0.000000    0.00           N/C
SCHEMA NAME          DSNP                 NOT ACCOUNTED                N/C    ARCHIVE LOG READ         0.000000    0.00           N/C
SUCC AUTH CHECK                       0   AVG.DB2 ENTRY/EXIT        672.48    DRAIN LOCK               0.000000    0.00           N/C
OCCURRENCES                          25   DB2 ENTRY/EXIT             16812    CLAIM RELEASE            0.000000    0.00           N/C
NBR OF ALLOCATIONS                 8406                                       PAGE LATCH               0.000000    0.00           N/C
SQL STMT - AVERAGE               336.24   CP CPU SU                  58.60    NOTIFY MESSAGES          0.000000    0.00           N/C
SQL STMT - TOTAL                   8406     AGENT                    58.60    GLOBAL CONTENTION        0.000000    0.00           N/C
NBR RLUP THREADS                     25     PAR.TASKS                 0.00    TCP/IP LOB XML           0.000000    0.00           N/C
                                          SE CPU SU                 100.78    ACCELERATOR              0.000000    0.00           N/C
                                                                              TOTAL CL8 SUSPENS.       0.000000    0.00
```

12.6 Hardware Instrumentation Services

When a performance problem is encountered, IBM might instruct the customer to enable the CPU measurement facility, record the data in a file, and send the data to IBM for analysis. Starting with IBM System z10® there is a new hardware instrumentation facility, the *CPU Measurement Facility* (CPU MF), which provides support built into the processor to aid in capacity planning and performance analysis. There is also a z/OS component *Instrumentation*, or *Hardware Instrumentation Services* (HIS) to support the CPU MF. Using this mechanism allows customers to observe the performance behavior of an operating system running client applications with little impact to the system being observed.

CPU MF does not replace existing performance monitoring capabilities, but provides an enrichment of the existing capabilities. Because CPU MF sampling has a low overhead, assuming that appropriate sampling intervals are used, it is possible to collect millions of samples over a relatively short interval, meaning that the activity of even little-used code can be observed. Also, because samples can be taken even when the CPU is not enabled for interrupts, it is possible to gather information about processing that normally cannot be monitored easily by traditional DB2 traces.

In addition to performance problem diagnosis, the CPU MF can potentially be used by IBM to provide the tuning guidance to the customer's workload. For example, it can provide translation lookaside buffer (TLB: used for translating virtual address to real address) miss rate, which can be used to understand the benefit of using large page frames.

The CPU MF data supplements the existing performance metrics that are traditionally collected using SMF, RMF, and so on.

12.6.1 CPU Measurement Facility

The CPU Measurement Facility consists of two parts:

- A CPU Measurement Counter Facility

  This facility provides a means to measure activities in the processor and in various shared peripheral processors. Using HIS, you can capture counter data indicating the frequencies of various hardware events such as cycles, instructions, and cache misses. The counters provide detailed information that can be used for capacity planning as well as performance analysis. There is negligible overhead in collecting the counters data. It is recommended to collect and save the counters data (as SMF type 113 records) in production environments on an on-going basis.

- A CPU Measurement Sampling Facility

  This facility provides a means to take a snapshot of the processor at a specified sampling interval. The facility can collect instruction samples that contain all the information needed to determine the address space and address where the sampled instruction is running. The collection and analysis of the sampling data is usually done under guidance of IBM technical personnel to resolve specific issues. Customers need not collect the sampling data on an on-going basis.

12.6.2 Hardware Instrumentation Services

CPU MF is the hardware side of the data collection process. The software that is responsible for retrieving the data from CPU MF, and controlling CPU MF, is the HIS component of z/OS. HIS was integrated into z/OS 1.11 and delivered back to z/OS 1.8 with an APAR.

Details about setting up the HIS and usage scenarios can be found in *Setting Up and Using the IBM System z CPU Measurement Facility with z/OS*, REDP-4727.

A

Recent maintenance

These APARs represent a snapshot of the current maintenance at the time of writing. For an up-to-date list, ensure that you contact your IBM Service Representative for the most current maintenance at the time of your installation. Also check on IBM RETAIN for the applicability of these APARs to your environment and to verify prerequisites and post-requisites.

Use the Consolidated Service Test (CST) as the base for service as described at the following site:

`http://www.ibm.com/systems/z/os/zos/support/servicetest`

DB2 11 and DB2 10 are included in the RSU.

Testing for RSU service package RSU1406 (CST2Q14) is now complete. This RSU completes the first quarter of 2014. This July 2014 2nd Quarter 2014 2nd Addendum “Quarterly Report” (109 KB PDF file) contains all service through the end of March 2014 not already marked RSU.

This service also includes PE resolution and HIPER/Security/Integrity/Pervasive PTFs and their associated requisites and supersedes through May 2014, as described at:

`http://www.ibm.com/systems/resources/RSU1406.pdf`

A.1 DB2 APARs

Table A-1 lists the APARs that provide functional and performance enhancements to DB2 11 for z/OS. This list is not and cannot be exhaustive; check **RETAIN** and the DB2 website for a complete list.

Table A-1 DB2 10 current function and performance related APARs

APAR #	Area	Text	PTF and notes
II10817	Storage	Info APAR for storage usage error	
II11334	TCP/IP	Info APAR for Communication Server	
II14219	zIIP	zIIP exploitation *support use* information	
II14334	LOBs	Info APAR to link together all the LOB support delivery APARs	
II14426	XML	Info APAR to link together all the XML support delivery APARs	
II14441	Incorrout PTFs	Preferred DB2 9 SQL INCORROUT PTFs	
II14587	Work file	DB2 9 and 10 work file recommendations	
II14619	Migration	Info APAR for DB2 10 DDF migration	
II14660	V11 migration	Info APAR to link together all the migration APARs to V11	
PI09807	-STOP ACCEL	-stop accelerator command may hang the subsystem	UI16997
PI09336	Statistics	Incorrect skip reading MCARDS SYSKEYTGTDIST frequency or histogram statistics exist in SYSCOLDIST for index on expression	UI15908
PI08339	REORG	Automated building of mapping tables	OPEN
PI08151	RUNSTATS	Filter factor too high for join predicate with histogram statistics on join column resulting in bad access path OPEN	UI15313
PI07794	RUNSTATS	sysstatfeedback dsn_stat_feedback reason conflict due to runstats collecting wrong cardinality for multicolumn mcard	UI18721
PI07694	Log apply	List prefetch for Fast Log Apply (FLA) implemented for RECOVER INDEX	UI14004 also V9, V10
PI06695	Access path	Bad access path due to incorrect filter factor for range or between predicate with timestamp column	UI14644
PI06690	DBD01	Reduce space map contention for lob ts	OPEN
PI06540	SINGLETON SELECT	High locking activity for singleton selects with iso(cs) and currentdata(yes)	UI16084
PI06261	LOG	Log read through ifi access to quiesced member's log and bsds. CLOSED	UI13582 also V9, V10

APAR #	Area	Text	PTF and notes
PI06018	RENAME TABLE	Performance degradation in rename table	UI13776 also V10
PI04769	Access path	Different access path is selected with production modeling because index bp size does not honor profile setting	UI14001
PM99792	UTS insert	Excessive page searches for pbg & pbr uts under DB2 10 NFM	UI14264 also V10
PM98832	Work file	Support in-memory Group By sort	UI13038 V10
PM98683	Index on expression	DB2 10 and 11 not able to match expressions using an index on expression created in DB2 9 with a timestamp key	UI12626 also V10
PM98330	Inner join	Complex query results in long running/hung thread	UI14054
PM97868	REORG	Excessive elapsed time on mapping index build during REORG execution	UK98424 also V9, V10
PM97845	XML	XML Index using sequential prefetch instead of dynamic prefetch	UI13152 also V10
PM97744	Access path	Unnecessary sort for an SQL with an inlist GROUP BY and ORDER BY	UI14128
PM96958	Access path	Incorrect index probing during prepare for equal and in predicates on a column with high colcardf	UI12702
PM96672	LOAD	DB2 11 load utility ran into hang situation	UK97615
PM96669	Access path	APREUSE failure when the DPSI could support the order by order and there is only one part qualified	UI12325
PM96334	Access path	Join predicate for page range screening is not effective because an internal variable is not set correctly	UI12250
PM96004	Thread break in	Enabling code	UI12985
PM96001	Thread break in	Toleration code for all V11 members	UI12985
PM95929	Thread break in	The need to break in for BIND/DDL activity - Early code	UI13368
PM95689	Access path	Inaccurate filter factor when single-column histogram stats are generated by index probing but not cover complete column dist	UI13054 also V10
PM95541	Incorrect values	Forward fit of pm82908 predicate that references table function input parameter is stage 2	UK98205
PM95294	ALTER	Reduce sync getpages against DSNTPX01 index of SYSCOLDISTSTATS table	UK97912 also V10
PM95294	Statistics	Alter table drop column has adverse performance when scanning sysibm.syscoldiststats	UK97912 also V10
PM95113	Access path	Enable subquery to join transformation for subqueries with correlation predicates on columns defined with a fieldproc	UI12406 also V10

APAR #	Area	Text	PTF and notes
PM94967	Access path	Inaccurate filter factor estimation for in-list predicates when index probing is applicable	UK98113
PM94911	Access path	Performance regression on partition table with equal predicate on timestamp column (page_range = y) after migration to DB2 10	UI11781 also V10
PM94911	Access path	See PM73542	UI11781 also V10
PM94815	Access path	Unnecessary sort is used when reverse index scan can avoid it	UK98529 also V10
PM94767	Access path	Optimizer does not use a non-matching index access to avoid a sort with optimize for one row	UK98034 also V10
PM94715	ENFM	Improve step ENFM001. Systems with a large number of rows in SYSIBM.SYSCOLUMNS the DSNTIJEN step ENFM0001 can take longer to complete	UK97335
PM94681	ADMIN_INFO_SQL	Collection features, enhancements, and service	UI19546 also V10
PM94525	Access path	Inefficient access path may occur due to overweighted I/O estimate	UK98039
PM94330	XML	Internal xml control block is under utilizing allocated storage	UI12940 also V10
PM94248	Access path	An inaccurate filter factor may be produced for a predicate with varchar/vargraphic/varbinary column when index probing is used	UK97779 also V10
PM94160	Join	DB2 may select an inefficient join sequence for a query with multiple tables joins	UK98124 also V10
PM94081	LIKE or BETWEEN	Inefficient access path is chosen when the query contains like or between predicate due to wrong filter factor estimation	UK98289 also V10
PM93976	IS NULL	Various abends or poor query performance for a query with a is null predicate on a column with a single histogram quantile	UK97012
PM93690	Accelerator	Analytics Accelerator related dbats require transaction accounting records and should not be susceptible to idle thread timeout conditions	UK97253
PM93688	PBG	DB2 11 nfm: partition-by-growth table space with member cluster gets high getpage count during highly intensive insert workload	UK97592 also V10
PM93577	Query in DSNESQ	DSNTESQ insert INS32 needs to be updated	UK98216 also V9, V10
PM93375	CASTOUT	Elapsed time degradation and increased sync I/O in parallel index updates	UK97639
PM93239	MODIFY	Modify at part level performance issue	UK97766
PM92730	DSNTIJMV	Corrections to job for migration	UK98196

APAR #	Area	Text	PTF and notes
PM92594	Predicate push down	Problems related to the predicate pushdown that may cause performance issue and incorrect plan table	UK97329
PM92517	Inefficient index	A suboptimal index may be chosen due to optimizer internal cost reduction issue that can result in an inefficient access path	UK96884
PM92272	BETWEEN on TIMESTAMP	Between predicate on a timestamp column under an or not eligible for multi-index access	UK9747
PM91565	Premigration DSNTIJPM	SQLCODE -104	UK95419
PM91514	Data sharing	Increase in sync I/O on lobs in a data sharing environment	UK96629 also V10
PM91162	IFC	Very high DB2 cl2 cpu time in accounting records after a tran with udf did rollback	UI13613 also V9, V10
PM90987	DPSI	Index with less matchcols chosen instead of dpsi with a superset matchcols when there is only one part qualified. CLOSED	UK96986 also V10
PM90422	IFC enhancements	ifc enhancements (IFCID 199 and others)	UI13998 also V9, V10
PM90188	Access Path	Wrong access path is used when there is exists predicate in the select list and predicate transitive closure is triggered	UK96470 also V10
PM89655	DSNZPARM	Restrictions for IXcontrolled TS PREVENT_NEW_IXCTRL_PART and PREVENT_ALTERTB_LIMITKEY	UK98188 also V10
PM89117	V11 Migration	New functions	UK95677 V10
PM89016	Locking	Reintroduce lock avoidance for non-PBG tables when fully matching index columns	UI12870 V9 and V10
PM85053	IRLM	IRLM enhancement for DB2 11 to suppress unnecessary child lock propagation to the CF lock structure	UK92783
PM84765	IRLM	New option (QWAITER) to QUERYFST request used by DB2	UK92494
PM83690	IRLM	Reduce IRLM latch contention and CPU time when there is a large number of parent locks per thread as in release deallocate	UK93840 V10
PM82279	Data sharing	Synchronous log write in Update of GBP-dependent data resulting in an overflow	UK93522 V10
PM80004	DDF	Synchronous Receive	UK92097
PM73542	Access path	Multi-index access not used with a predicate comparing a timestamp column to a literal value. See also PM94911	UI13819 also V10
PM45652	Migration	prefix.SDSNLINK lib	UK74535 V10

APAR #	Area	Text	PTF and notes
PM31841	V11 Migration	Toleration of fall back to DB2 10	UK96357 V10

A.2 z/OS APARs

Table A-2 lists the APARs that provide additional enhancements for z/OS. This list is not and cannot be exhaustive; check **RETAIN** and the DB2 website for a complete list.

Table A-2 z/OS DB2 related APARs

APAR #	Area	Text	PTF and notes
OA37550	Coupling Facility	Performance improvements are needed for coupling facility cache structures to avoid flooding the coupling facility cache with changed data and avoid excessive delays and backlogs for cast-out processing	UA66419
OA39392	New function	Terminate a pre-emptible SRB in the -CANCEL THREAD with FORCE option	UA66823
OA40967	RSM Enablement Offering	2 GB frame support	UA68170
OA41617	DFSMS	DFSMS control block accessing support for `NON_VSAM_XTIOT = YES` in DEVSUPxx	UA69320

A.3 OMEGAMON PE APARs

Table A-3 lists the APARs that provide additional enhancements for IBM Tivoli OMEGAMON XE for DB2 PE on z/OS V5.2.0, PID 5655-W37. This list is not and cannot be exhaustive; check **RETAIN** and the DB2 tools website for a complete list.

Table A-3 OMEGAMON PE GA and DB2 10 related APARs

APAR #	Area	Text	PTF and notes
II14438		Info APAR for known issues causing high CPU utilization	

A.4 To REBIND or not to REBIND

In this section, we summarize the major DB2 11 functions by binding and other requirements.

A.4.1 Performance enhancements: No REBIND needed (conversion mode (CM))

- DDF performance improvements:
  - Reduced SRB scheduling on TCP/IP receive using new CommServer capabilities
  - Improved autocommit OLTP performance

- xProcs above the bar
- zIIP enablement for all SRB-mode DB2 system agents that are not response time critical
- Avoid cross-memory overhead for writing log records
- Data decompression performance improvement
- INSERT performance:
  - Latch contention reduction
  - CPU reduction for Insert column processing and log record creation
  - Data sharing LRSN spin avoidance
  - Page fix/free avoidance in GBP write
- Sort performance improvements
- DPSI performance improvements for merge
- Performance improvements with large number of partitions
- XML performance improvements
- Optimize RELEASE(DEALLOCATE) execution so that it is consistently better performing than RELEASE(COMMIT)
- IFI 306 filtering capabilities to improve Replication capture performance
- Utilities performance improvements
- Automatic index pseudo-delete clean-up
- ODBC/JDBC Type 2 performance improvements
- Java stored procedures:

  Multi-threaded JVMs, 64-bit JVM: more efficient
- ACCESS DATABASE command performance
- Declared global temporary table (DGTT) performance improvement:

  Avoid incremental binds for reduced CPU overhead
- P-procs for LIKE predicates against Unicode tables
- Improved performance for ROLLBACK TO SAVEPOINT
- zEC12 exploitation
- Latch contention reduction and other high n-way scalability improvements
- Data sharing performance improvements

A.4.2 Performance enhancements requiring REBIND (CM with or without REUSE)

- Most in-memory techniques
- Non-correlated subquery with mismatched length
- Select list do-once
- Column processing improvements
- RID overflow to work file handled for Data Manager set functions
- Performance improvements for common operators
- DECFLOAT data type performance improvements
- Query transformation improvements: less expertise required to write performant SQL
- Enhanced duplicate removal
- DPSI and page range performance improvements
- Optimizer CPU and I/O cost balancing improvements

A.4.3 Performance enhancements: DBA or application effort required (NFM)

- Suppress-null indexes
- New PCTFREE FOR UPDATE attribute to reduce indirect references
- DGTT performance improvements
- Global variables
- Optimizer externalization of missing and conflicting statistics
- Extended optimization: selectivity overrides (filter factor hints)
- Open data set limit raised to 200 K

A.4.4 Optional enhancements need NFM + DBA effort

- DSNTIJCB – Optional – Convert BSDS for extended 10-byte RBAs:
  - - STOP DB2 MODE(QUIESCE)
- DSNTIJCV – Optional – Convert Catalog and Directory table and index spaces to extended 10-byte RBA format:
  - Reorgs all Catalog and Directory table spaces SHRLEVEL CHANGE
  - Can be split up to run reorgs in parallel

B

The IBM DB2 workloads

This appendix provides an overview of the workloads used to measure DB2 performance.

Traditionally, performance workloads started as heavy-duty batch jobs, which stressed CPU and I/O of the system.

With the advent of teleprocessing and multi-address spaces, online TP became necessary and often terminal simulators have been used. At DB2 Version 4 time the online TP IBM Relational Warehouse Workload (IRWW) was created, based on a retail type of environment and using IMS and later CICS as transaction monitor.

In later years, more workloads came up related to complex online TP, distributed environments, query, and data warehousing.

Special high insert batch workloads were designed to compare physical design options and different structures to minimize contention.

From Chapter 1:

- A set of IRWW workloads represents the simple, legacy type, OLTP workload typically executed through IMS or CICS as transaction server.
- Distributed IRWWs are the same DB2 workloads executed through remote clients, such as JDBC or remote stored procedure calls.
- Brokerage OLTP represents a complex OLTP with a mixture of look up, update, and insert as well as reporting queries.
- SAP Banking Day Posting workloads are relatively simple SQL executions against an SAP banking schema.
- High insert workloads are concurrent inserts executed through more than 100 JCC T4 connections in data sharing.
- TSO batches are representing a batch workload executing various batch operations including SELECT, FETCH, UPDATE, INSERT, and DELETE in a loop.
- A set of query workloads are executed serially mostly using CPU parallelism. Each query workload consists of a set of 20 to 100 queries per workload. Individual queries can see wide range of differences between DB2 10 and DB2 11. The average CPU reduction rate is shown here.

- Cognos BI-day short workload is a set of short running reporting queries concurrently executed.
- Utilities scenarios are a set of various utility job executions.
- XML scenarios are also a set of various XML workloads including OLTP, queries, batch, and utilities against XML data types.
- WebSphere Application Server Portal workload is used to illustrate this process, and the performance impacts that this can produce.

In this appendix we briefly describe:

- IBM Relational Warehouse Workload
- Distributed IRWW workload description (more details in 9.8, "Distributed IRWW workload measurements" on page 225)
- Brokerage workload
- SAP Banking Day Posting workload

B.1 IBM Relational Warehouse Workload

IRWW is an OLTP workload that consists of seven transactions. Each transaction consists of one to many SQL statements, each performing a distinct business function in a predefined mix.

Table B-1 provides a summary of the transactions used in the Classic IRWW Workload.

Table B-1 Classic IRWW transactions

Transaction	Relative % mix	DML executed
Delivery	2	OPEN/FETCH/SELECT/UPDATE/DELETE
New Order	22	OPEN/FETCH/SELECT/UPDATE/INSERT
Order Status	24	OPEN/FETCH/SELECT/
Payment	22	UPDATE/INSERT
Price Change	1	UPDATE
Price Quote	25	SELECT
Stock Inquiry	4	SELECT

The seven transaction types, a brief description of each, and the percentage of the transaction mix follows:

- Neworder

  Performs various SELECTS, FETCHES, UPDATES, and INSERTS in support of the receipt of new customer orders and runs as 22% of the total transaction mix.

- Order Status

  Performs various SELECTS and FETCHES in support of providing the status of an order and runs as 24% of the total transaction mix.

- Payment

  Performs SELECTS, FETCHES, UPDATES, and INSERTS in support of received customer payments and runs at 22% of the total transaction mix.

- Price Change

  Performs an UPDATE in support of changing the price of an item and runs as 1% of the total transaction mix.

- Price Quote

  Performs various SELECTS in support of providing the price of a set of items and runs as 25% of the total transaction mix.

- Stock Level

  Performs a JOIN and various SELECTS in support of providing the current stock level of an item and runs at 4% of the mix.

- Delivery

  Performs various SELECTS, UPDATES, and DELETES in support of the delivery of a group of orders and runs as 2% of the total transaction mix.

The IRWW database contains a variable number of inventory stock warehouses and sales districts.

The front end, transaction manager is usually IMS but sometimes CICS is used.

B.2 IRWW distributed workload

The IRWW OLTP workload consisting of seven transactions in support of a product warehouse, customer order, and order delivery system, was converted to run in distributed environments. Each transaction consists of roughly 25 DML on average with seven of those being Insert/Update/Delete. The workloads consist of stored procedure and non-stored procedure workloads.

The seven transaction types, a brief description of each, and the percentage of the transaction mix can be found in the preceding section, “IBM Relational Warehouse Workload”.

See 9.8, “Distributed IRWW workload measurements” on page 225 for the measurements.

B.3 Brokerage

Brokerage OLTP represents a complex OLTP with a mixture of look up, update, and insert as well as reporting queries.

The IBM Brokerage transaction workload is a complex OLTP workload, which executes various SQL procedures to simulate transactions in a brokerage farm.

An OLTP workload was run that accesses 33 tables with about 15,000 objects. The Brokerage workload represents transactions of customers dealing in the financial market and updating account information. The transactions are rather complex OLTP transactions implemented in stored procedure using SQL/PL:

- 1.3 TB of data

- 13 buffer pools
- 33 table spaces
- 68 indexes

B.4 SAP Banking Day Posting

SAP Banking Day Posting workloads are relatively simple SQL executions against an SAP banking schema.

High insert workloads are concurrent inserts executed through more than 100 JCC T4 connections in data sharing.

TSO batches are representing a batch workload executing various batch operations including SELECT, FETCH, UPDATE, INSERT, and DELETE in a loop.

C

To REBIND or not to REBIND

DB2 11 is another version of DB2 for z/OS that is focused on performance and innovative features. There are many enhancements that will invite you start planning soon for migration.

Plan carefully and you will get the best out of the new DB2 features. For static SQL applications, you are likely to have to REBIND packages to activate the improvements. To help you plan a safe REBIND strategy, you can consider using REBIND with APREUSE and APCOMPARE.

To help you plan for REBIND, we summarize in this Appendix the DB2 11 functions trying to clarify if they are available in NFM or conversion mode (CM), and if rebinding or other requisite actions are needed.

This is a classification that is subject to errors due to possible changes via maintenance.

C.1 Performance enhancements - no REBIND needed (CM)

- DDF performance improvements:
  - Reduced SRB scheduling on TCP/IP receive using new CommServer capabilities
  - Improved autocommit OLTP performance
- xProcs above the bar
- zIIP enablement for all SRB-mode DB2 system agents that are not response time critical
- Avoid cross-memory overhead for writing log records
- Data decompression performance improvement
- INSERT performance:
  - Latch contention reduction
  - CPU reduction for Insert column processing and log record creation
  - Data sharing LRSN spin avoidance
  - Page fix/free avoidance in GBP write
- Sort performance improvements
- DPSI performance improvements for merge
- Performance improvements with a large number of partitions
- XML performance improvements
- Optimize RELEASE(DEALLOCATE) execution so that it is consistently better performing than RELEASE(COMMIT)
- IFI 306 filtering capabilities to improve Replication capture performance
- Utilities performance improvements
- Automatic index pseudo-delete clean-up
- ODBC/JDBC Type 2 performance improvements
- Java stored procedures:

  Multi-threaded JVMs, 64-bit JVM: more efficient
- ACCESS DATABASE command performance
- Declared global temporary table (DGTT) performance improvement:

  Avoid incremental binds for reduced CPU overhead
- P-procs for LIKE predicates against Unicode tables
- Improved performance for ROLLBACK TO SAVEPOINT
- zEC12 exploitation:
  - Pageable 1 MB frames
  - 2 GB frames
  - 1 MB frames for code (z/OS 2.1 and Flash Express)
- Latch contention reduction and other high n-way scalability improvements
- Data sharing performance improvements:
  - Spin reduction with extended LRSN
  - Castout performance
  - GBP write-around
  - Index split performance

C.2 Performance enhancements requiring REBIND (CM with or without REUSE)

- Most In-memory techniques:
  - In-memory, reusable workfiles
  - Sparse index hashing
  - Non-correlated subquery using MXDTCACH
  - Correlated subquery caching
- Non-correlated subquery with mismatched length
- Select list do-once:

  Non-column expressions in the select list can be executed once (versus per row)
- Column processing improvements
- Record identifier (RID) overflow to work file handled for Data Manager set functions
- Performance improvements for common operators:

  MOVE, CAST, CASE, SUBSTR, DATE, output hostvar processing
- DECFLOAT data type performance improvements
- Query transformation improvements: less expertise required to write performant SQL
- Enhanced duplicate removal
- DPSI and page range performance improvements
- Optimizer CPU and I/O cost balancing improvements

C.3 Performance enhancements: DBA or application effort required (NFM)

- Suppress-null indexes
- New PCTFREE FOR UPDATE attribute to reduce indirect references
- DGTT performance improvements
- Global variables
- Optimizer externalization of missing/conflicting statistics
- Extended optimization: selectivity overrides for predicates (filter factor hints)
- Open data set limit raised to 200 K

C.4 Optional enhancements need NFM and DBA effort

- DSNTIJCB – Optional – Convert BSDS for extended 10-byte RBAs:
  - - STOP DB2 MODE(QUIESCE)
- DSNTIJCV – Optional – Convert Catalog and Directory table and index spaces to extended 10-byte RBA format:
  - Reorgs all Catalog and Directory table spaces SHRLEVEL CHANGE
  - Can be split up to run reorgs in parallel

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide additional information about the topic in this document. Note that some publications referenced in this list might be available in softcopy only.

- *DB2 11 for z/OS Buffer Pool Monitoring and Tuning*, REDP-5092
- *DB2 11 for z/OS Technical Overview,* SG24-8180
- *Subsystem and Transaction Monitoring and Tuning with DB2 11 for z/OS*, SG24-8182
- *DB2 10 for z/OS Performance Topics,* SG24-7942
- *DB2 10 for z/OS Technical Overview*, SG24-7892
- *DB2 10 for z/OS: Configuring SSL for Secure Client-Server Communications,* REDP-4799
- *Setting Up and Using the IBM System z CPU Measurement Facility with z/OS*, REDP-4727
- *DB2 for z/OS and List Prefetch Optimizer,* REDP-4862
- *IBM System z Connectivity Handbook*, SG24-5444-13
- *DB2 UDB for z/OS Version 8: Everything You Ever Wanted to Know, ... and More*, SG24-6079

You can search for, view, download or order these documents and other Redbooks, Redpapers, Web Docs, draft and additional materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- *DB2 11 for z/OS Administration Guide*, SC19-4050
- *DB2 11 for z/OS Application Programming and SQL Guide*, SC19-4051
- *DB2 11 for z/OS Application Programming Guide and Reference for Java*, SC19-4052
- *DB2 11 for z/OS Codes*, GC19-4053
- *DB2 11 for z/OS Command Reference*, SC19-4054
- *DB2 11 for z/OS Data Sharing: Planning and Administration*, SC19-4055
- *DB2 11 for z/OS Installation and Migration*, SC19-4056
- *DB2 11 for z/OS Internationalization Guide*, SC19-4057
- *Introduction to DB2 for z/OS*, SC19-4058

- *DB2 11 for z/OS DB2 11 for z/OS IRLM Messages and Codes for IMS and DB2 for z/OS,* GC19-2666
- *DB2 11 for z/OS Managing Performance*, SC19-4060
- *DB2 11 for z/OS Managing Security*, SC19-4061
- *DB2 11 for z/OS Messages*, GC19-4062
- *DB2 11 for z/OS ODBC Guide and Reference*, SC19-4063
- *DB2 11 for z/OS pureXML Guide*, SC19-4064
- *DB2 11 for z/OS RACF Access Control Module Guide*, SC19-4065
- *DB2 11 for z/OS SQL Reference*, SC19-4066
- *DB2 11 for z/OS Utility Guide and Reference*, SC19-4067
- *DB2 11 for z/OS What's New?*, GC19-4068
- *DB2 11 for z/OS Diagnosis Guide and Reference,* LY37-3222
- *IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS V520 Report Reference,* SH12-6991

Online resources

These websites are also relevant as further information sources:

- DB2 11 for z/OS link

  https://www.ibm.com/software/data/db2/zos/family/db211
- How to Benefit from Hardware Instrumentation Services Data

  http://enterprisesystemsmedia.com/article/how-to-benefit-from-hardware-instrumentation-services-data
- SMC-R RoCE FAQ Document

  http://www-01.ibm.com/support/docview.wss?uid=tss1fq131485&aid=1
- Performance and Capacity Considerations for zIIP Engines:

  http://ibmdatamag.com/2013/04/performance-and-capacity-considerations-for-ziip-engines
- DB2 for z/OS Information Roadmap
  http://www.ibm.com/support/docview.wss?uid=swg27039818
- Migration planning workshop

  http://ibm.co/IIJxw8

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Q

R

S

IBM DB2 11 for z/OS Performance Topics

(0.5” spine)
0.475”<->0.873”
250 <-> 459 pages

IBM DB2 11 for z/OS Performance Topics

Reduce processor time in CM and NFM

Improve scalability and availability

Evaluate impact of new functions

IBM DB2 10 for z/OS provided a strong performance improvement that drove great value for IBM clients. DB2 11 for z/OS continues the trend by focusing on further CPU savings and performance improvement through innovations within DB2 and the synergy with IBM System z hardware and software. Most of CPU reductions are built in directly to DB2, requiring no application changes. Enhancements often take effect after the REBIND in the case of static applications, while some enhancements require user actions such as Data Definition Language (DDL) updates.

DB2 11 CPU reduction can potentially provide significant total cost of ownership savings depending on the application mixture and type of transactions.

In this IBM Redbooks publication we provide the details of the performance of DB2 11 for z/OS discussing the possible performance impacts of the most important functions when migrating from DB2 10. We include performance measurements made in the Silicon Valley Laboratory and provide considerations and high-level estimates. Keep in mind that results are likely to vary because the conditions and work will differ.

In this book, we assume that you are somewhat familiar with DB2 11 for z/OS. See DB2 11 Technical Overview, SG24-8180, for an introduction to the new functions.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-8222-00 ISBN 0738439673