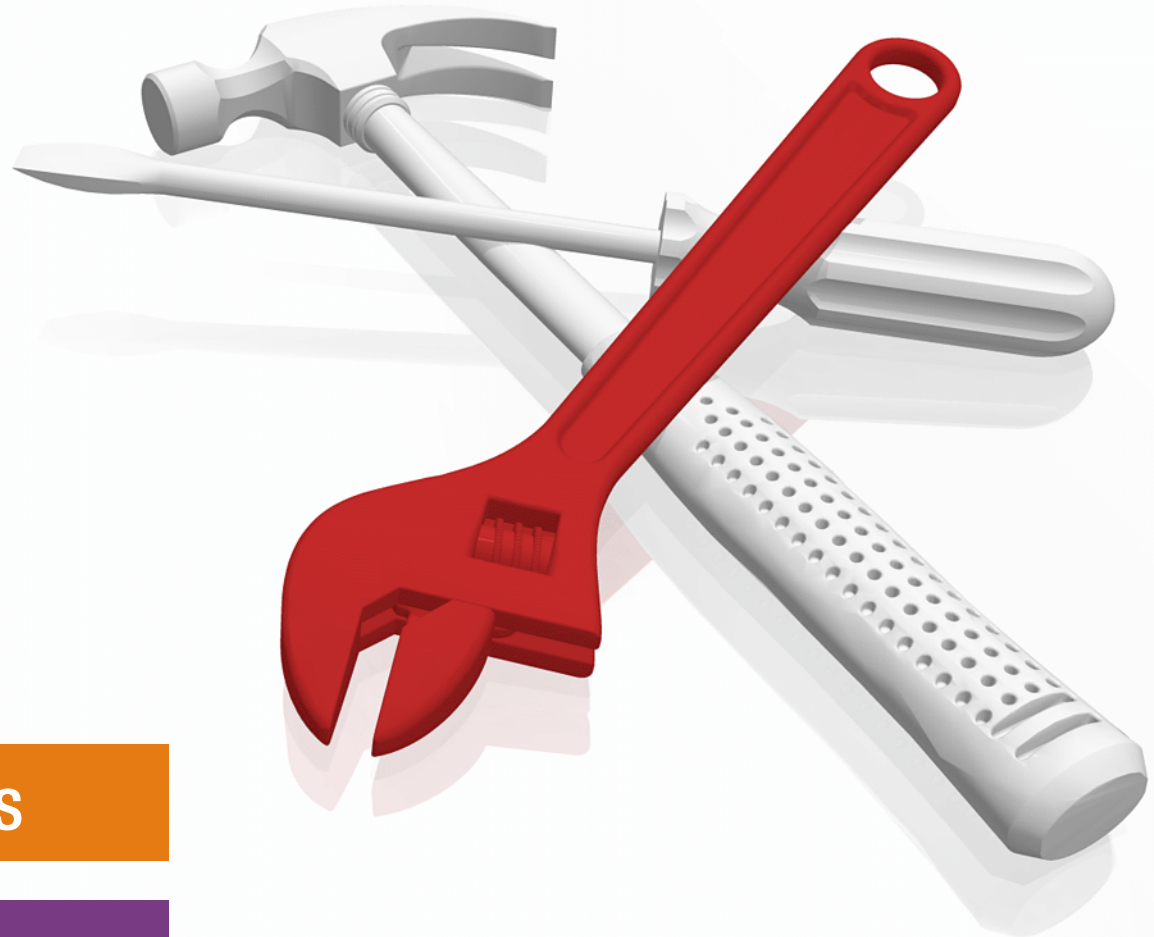


Db2 Optimization Techniques for SAP Database Migration to the Cloud

Dino Quintero
Frank Becker
Holger Hellmuth
Joern Klauke
Thomas Rech
Alexander Seelert
Tim Simon
Hans-Jürgen Zeltwanger



 Analytics

Information Management



IBM Redbooks

**Db2 Optimization Techniques for SAP Database
Migration to the Cloud**

March 2024

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

First Edition (March 2024)

This edition applies to the following software and operating system levels:

AIX 7.2 TL 5 SP 3

Db2 Version 11.1 MP4 FP6 SAP4

Db2 Version 11.5 MP7 FP0 SAP2

SAP ERP 6.08 (BS7i2016), kernel release 753

Red Hat Enterprise Linux 8.4

Red Hat Enterprise Linux 8.5

© Copyright International Business Machines Corporation 2024. All rights reserved.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	xi
Trademarks	xii
.....	
.....	Preface
Authors	xiii
Now you can become a published author, too!	xv
Comments welcome	xv
Stay connected to IBM Redbooks	xvi
Chapter 1. Introduction	1
1.1 Overview	2
1.2 SAP NetWeaver and Db2 in the cloud	2
1.3 Methods outside the scope of this document	4
1.4 Heterogeneous system copy	4
1.5 Unicode conversion	5
1.6 Database vendor change	5
Chapter 2. Db2 with SAP in the Cloud	23
2.1 SAP in the cloud	24
2.2 Db2 features for cloud deployments	24
2.3 Sizing of SAP systems for cloud service providers	25
2.4 Storage considerations	26
Chapter 3. Migration essentials	29
3.1 Introduction	30
3.2 Top recommendations	30
3.2.1 General recommendations	30
3.2.2 Standard recommendations	31
3.2.3 Five advanced recommendations	33
3.2.4 More recommendations	34
3.3 Resource usage during the migration	34
3.3.1 Using your resources	35
3.3.2 Ensuring the constant usage of resources	36
3.3.3 Balancing the resources	36
3.3.4 Achieving the best overall throughput	37
3.3.5 Avoiding unnecessary steps, tasks, and resource consumption	38
3.3.6 Using the most suitable resources	38
3.4 Using the SAP migration monitor	41
3.4.1 Configuring monitors for optimization	41
3.4.2 Using multiple monitor instances	42
Chapter 4. Tools overview	43
4.1 SAP Software Provisioning Manager	44
4.2 Embedded load tools	44
4.2.1 R3load	44
4.2.2 R3ldctl	45
4.2.3 R3szchk	46
4.3 Migration monitor	46
4.4 Time analyzer (MIGTIME)	47

4.4.1	Export times analysis	47
4.4.2	Import times analysis	49
4.4.3	Time join analysis	52
4.5	Tools for package and table splitting	53
4.6	SAP Migration Checker	54
4.7	SAP data definition tools	56
4.7.1	SMIGR_CHECK_DB6	56
4.7.2	SMIGR_CREATE_DDL	58
Chapter 5. Export optimization techniques		65
5.1	Activities to be done before export	66
5.2	Unsorted versus sorted export	66
5.2.1	Runtime comparison	67
5.2.2	Enabling unsorted export	67
5.2.3	Resource consumption	68
5.2.4	Sorting and cluster tables	69
5.2.5	Table clusters and cluster tables	69
5.3	R3load export dump compression	71
5.3.1	Table and dump-file size comparison	71
5.3.2	Static R3load compression	72
5.3.3	Adaptive R3load compression	72
5.3.4	Improvements with adaptive R3load compression	72
5.3.5	Usage of adaptive R3load compression	74
5.4	Package splitting	74
5.4.1	SAP data classes	74
5.4.2	Package Splitter tool	75
5.5	Local or remote export server scenarios	76
5.5.1	Exporting from the SAP source database server	76
5.5.2	Exporting from an SAP application server	77
5.5.3	Identifying candidates for dedicated application servers	78
5.5.4	Recommendation for dedicated application servers	78
5.6	Db2 query block prefetch feature	79
5.6.1	Feature description	79
5.6.2	Performance impact	79
5.6.3	Enabling Db2 query block prefetch	80
5.7	Db2 workload management	80
5.8	Other Db2 features	81
Chapter 6. Basic Db2 layout and configuration options		85
6.1	Planning your Db2 target system	86
6.2	Tablespace principles and configuration	86
6.3	Tablespace pools	87
6.3.1	Prerequisites and functionality of tablespace pools	88
6.3.2	Usage and recommendations of tablespace pools	89
6.4	General layout considerations	91
6.4.1	File system caching	91
6.4.2	Page size	91
6.4.3	Extent size	91
6.4.4	NUM_IOCLEANERS	92
6.4.5	Prefetch size	92
6.4.6	Db2 parallel I/O	92
6.4.7	Support for 4K sector size	93
6.4.8	Preallocation of tablespaces	93

6.4.9	Logging configuration	94
6.5	R3LOAD with Db2 compression	96
6.5.1	Introduction to Db2 compression	96
6.5.2	R3load compression options overview	97
6.5.3	Compression rates	102
6.5.4	Performance considerations	103
6.5.5	Tables with large binary objects	104
6.5.6	Conclusion	106
6.6	Import optimization	106
6.6.1	Db2 LOAD versus INSERT	107
6.7	Using and optimizing Db2 LOAD with R3load	110
6.7.1	Configuring the db2load API	110
6.7.2	DB6LOAD_CPU_PARALLELISM	111
6.7.3	Db2 utility heap and DB6LOAD_DATA_BUFFER_SIZE	112
6.7.4	DISK_PARALLELISM	114
6.7.5	Indexing mode	114
6.7.6	Preserving order of the data	114
6.7.7	Optimal number of R3Load processes	115
6.7.8	Cleaning up a failed Db2 LOAD	116
6.8	Order and configuration of index creation	117
6.8.1	Order of index creation	117
6.8.2	SORTHEAP	119
6.8.3	SHEAPTHRES_SHR	119
6.9	Db2 buffer pools	120
6.10	SMP parallelism for index creation	121
6.11	LOCKTIMEOUT	123
6.12	Optimizing statistics collection	123
6.12.1	Early enablement of Db2 automatic runstats	123
6.12.2	Manual runstats	124
6.12.3	The db2look command	125
6.13	Importing by use of a database server versus a remote client	126
6.14	Other Db2 features	127
6.14.1	Using self-tuning memory management	128
6.14.2	INTRA_PARALLEL	129
6.14.3	Changed pages threshold and alternate page cleaning	129
6.14.4	Alternate page cleaning	130
6.14.5	Spilled sorts to a RAM drive	130
6.14.6	Disabling database logging	130
6.14.7	Db2 native encryption	131
Chapter 7.	Import/Export overlap	133
7.1	Socket transfer and table split overview	134
7.2	Socket option	136
7.2.1	Comparing file and socket migration method	137
7.2.2	Migration Monitor (MigMon) configuration	137
7.2.3	Socket transfer considerations	139
7.3	Table splitting	139
7.3.1	Table splitting introduction	139
7.3.2	Export considerations	140
7.3.3	Import considerations	145
7.3.4	Conclusions	150
Chapter 8.	Data transfer options	151

8.1 IBM Cloud	152
8.2 Amazon AWS	155
8.3 Microsoft Azure	156
8.4 Google Cloud Platform	157
8.5 IBM Aspera for SAP migrations example	158
8.5.1 Setup description	158
8.5.2 Hardware and software environment	159
8.5.3 Aspera usage	159
8.5.4 SAP files to transfer	160
8.5.5 Shell script description	160
Chapter 9. Special cases	165
9.1 Special table types	166
9.1.1 ITC	166
9.1.2 MDC	167
9.1.3 Range partitioned tables	167
9.1.4 Column-organized tables – BLU Acceleration	168
9.1.5 Hash partitioned tables and database partitioning feature	171
9.2 SAP Business Warehouse based systems	173
9.3 Db2 pureScale feature	173
Chapter 10. Tips for monitoring	175
10.1 Db2 monitoring	176
10.1.1 db2top and dsmtop	176
10.1.2 dmctop	177
10.1.3 Db2 MONREPORT module	178
10.1.4 Lock wait situations	178
10.1.5 Reorganization status	179
10.1.6 Log space usage	180
10.1.7 Sorts	180
10.1.8 Utility Heap size	181
10.1.9 SQL statement analysis	181
10.2 Monitoring a single R3load process	182
10.2.1 The List Utilities command	182
10.2.2 Db2 diagnostic log file	182
10.2.3 The List History command	183
10.2.4 R3load log file	183
Chapter 11. Homogeneous system copy methods	185
11.1 Homogeneous system copy with SAP SWPM	186
11.2 Db2 HADR	186
11.2.1 Prerequisites and limitations	187
11.2.2 Procedure	187
11.2.3 Customer experiences for an HADR migration	189
11.2.4 HADR for heterogeneous system copy	189
11.3 IBM Db2 Shift	189
11.3.1 Prerequisites and limitations for the usage of Db2 Shift	190
11.3.2 Features of Db2 Shift	190
11.3.3 Preparing the target and source system	191
11.3.4 Offline move	191
11.3.5 HADR build and move	192
11.3.6 Air-gapped move by using the copy method	193
11.4 Downtime minimized methods	194
11.4.1 SAP Database Migration Option in the SAP Update Manager	194

11.4.2 IBM Data Replication - Change Data Capture Replication	194
Related publications	197
IBM Redbooks	197
Online resources	197
Help from IBM	198

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

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, MD-NC119, Armonk, NY 10504-1785, US

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 jurisdictions 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 provide in any way it believes appropriate without incurring any obligation to you.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

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.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

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 actual people or business enterprises 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. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Trademarks

IBM, the IBM logo, and [ibm.com](http://www.ibm.com) are trademarks or registered trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at “Copyright and trademark information” at <http://www.ibm.com/legal/copytrade.shtml>

The following terms are trademarks or registered trademarks of International Business Machines Corporation, and might also be trademarks or registered trademarks in other countries.

AIX®	IBM®	pureScale®
Aspera®	IBM Cloud®	Redbooks®
BLU Acceleration®	InfoSphere®	Redbooks (logo)  ®
Db2®	POWER®	z/OS®
FASP®	Power8®	zSystems™

The following terms are trademarks of other companies:

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, 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.

Red Hat, OpenShift, are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

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

For many years, SAP migrations have been a standard process. An increasing number of customers are changing their database software to IBM® Db2® for UNIX, Linux, and Windows or are moving their existing Db2 based infrastructure from on-premises into the cloud. When customers move to the cloud, a heterogeneous system copy is often needed because of a change in the underlying hardware architecture and operating system.

This book provides in-depth information about the best practices and recommendations for the source system database export, the advanced migration techniques, database layout and configuration, database import recommendations, SAP NetWeaver Business Warehouse, in addition to background information about Unicode. The book includes best practices in a single chapter that can be used as a quick reference for experienced migration consultants. It describes optimization strategies and best practices for migrating SAP systems to IBM Db2 for Linux, UNIX, and Windows. It is intended for experienced SAP migration experts and discusses IBM Db2 specific recommendations and best practices. It addresses advanced SAP migration techniques, considerations for database layout and tuning, and presents unique Db2 capabilities.

All techniques discussed within this book are based on extensive tests and experiences collected from countless migration projects. However, it is important to understand that some advanced optimizations described in this document might introduce risks to the overall process because of their complexity. Other optimizations might require changes to the production system. Therefore, use these features during migration only when the downtime window might not be large enough.

The authors want this book to be as helpful as possible. If you want to provide feedback on the recommendations or if you have suggestions or questions, you are welcome to contact the authors.

Authors

This book was produced by a team of specialists from around the world working at IBM Redbooks, Austin Center.

Dino Quintero is a Systems Technology Architect with IBM Redbooks®. He has 28 years of experience with IBM Power® technologies and solutions. Dino shares his technical computing passion and expertise by leading teams that develop technical content in the areas of enterprise continuous availability, enterprise systems management, high-performance computing (HPC), cloud computing, artificial intelligence (AI) (including machine and deep learning), and cognitive solutions. He is a Certified Open Group Distinguished Technical Specialist. Dino is formerly from the province of Chiriqui in Panama. Dino holds a Master of Computing Information Systems degree and a Bachelor of Science degree in Computer Science from Marist College.

Frank Becker joined IBM in 2001 as a student in business informatics. After his certification as an SAP Technology Consultant for OS/DB migration, he acquired extensive skills in homogeneous and heterogeneous SAP system copy projects around the world. This was followed by combined upgrade and unicode conversions projects as a technical analyst, coach for the SAP Basis and member of the project management team. He was deeply involved in proof of concepts regarding IBM advanced SAP system copy procedure using

IBM InfoSphere® Data Replication Change Data Capture for downtime minimization. He holds a degree in business informatics from the Berufsakademie Dresden, Germany.

Holger Hellmuth is a software engineer in the Db2 for LUW SAP Porting Team in Walldorf, Germany. Previously he worked in technical marketing and sales support for SAP on Db2 for Linux, UNIX, and Windows at the IBM SAP International Competence Center in Walldorf. He has about 30 years of experience with SAP and Db2. Holger has worked for IBM for 35 years. He holds a degree in physical engineering from the University of Applied Science in Heilbronn, Germany. His area of expertise includes Db2 for SAP Software, Db2 backup and restore, and the implementation of Db2 on modern hardware and software environments. He has written extensively on the values of Db2 LUW for SAP NetWeaver-based implementations.

Joern Klauke leads the SAP on Db2 LUW development team. He joined IBM in 2008 and for fourteen years gained experience as a software developer for SAP on Db2 for Linux, UNIX, and Windows at the IBM Research and Development Lab in Böblingen, Germany while developing the scripted interface and NX842 compression for Db2 backups and log archives. He also worked as a support analyst for Db2 Advanced Support. Joern holds a degree in Computer Science from the Martin-Luther-University of Halle, Germany.

Thomas Rech has been working in the Db2 area since 1996. Starting as a Db2 course instructor, he soon moved to the SAP on Db2 environment in 1997. He joined the SAP/IBM Db2 Development Team, followed by a role as technical sales consultant. He led several lighthouse projects like the first Db2/SAP implementation on HP-UX, the internal SAP implementation of Db2, and the world's largest SAP Business Warehouse. He took over the role of IT architect and team lead in the IBM SAP Db2 Center of Excellence, helping customers in critical situations and everything around Db2 and SAP. Thomas now works as IT architect in the Db2/SAP development team out of the IBM Research & Development Laboratory in Böblingen, Germany. He holds a degree in computer science from the University of Applied Sciences in Worms.

Alexander Seelert joined IBM in 1998 as a certified SAP Technology Consultant for OS/DB Migration. He acquired deep skills in project management, migrations, and upgrades for any kind of SAP applications and databases. Together with selected clients, he used advanced migration methods to migrate complex systems using the IBM Data Replication application and SAP with near zero downtime to different selected Cloud Hyperscalers. In 2022, he became an IBM Technical Sales Specialist. In this new role, he is providing more insight and individual solutions for Hybrid Cloud scenarios.

Hans-Jürgen Zeltwanger is a member of the SAP on Db2 for Linux, UNIX, and Windows development team at IBM Germany Research & Development. He joined IBM in 1997 and has more than 25 years of experience with SAP and Db2. Hans-Jürgen started as a developer for SAP Ready-to-Run systems, followed by nearly a decade as a pan-European technical sales consultant. As team lead in the IBM SAP Db2 Center of Excellence, Hans-Jürgen successfully planned and performed countless proofs of concept and SAP migration projects. For many years, he has been involved in the development of training material for using Db2 as the database for SAP NetWeaver-based products and is leading the development of the SAP on Db2 Learning Journey. Hans-Jürgen holds a degree in electrical engineering from the Baden-Württemberg Cooperative State University and a degree in industrial engineering from Pforzheim University.

Tim Simon is an IBM Redbooks Project Leader in Tulsa, Oklahoma, US. He has over 40 years of experience with IBM, primarily in a technical sales role working with customers to help them create IBM solutions to solve their business problems. He holds a BS degree in Math from Towson University in Maryland. He has worked with many IBM products and has

extensive experience creating customer solutions by using IBM Power, IBM Storage, and IBM zSystems™ throughout his career.

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

- ▶ Christian Brockhaus, International Team Leader for SAP OS/DB Migration Solutions, IBM Germany
- ▶ Phil Downey, Technical Product Manager, IBM Research Discovery Tooling Research, IBM Australia
- ▶ Karin Eberhart, SAP Db2 LUW Installation Tools, Data and AI Software, IBM Germany
- ▶ Michael Hoffmann, BTS Automation Integration DACH Global Sales - Cloud Platform Sales, IBM Germany
- ▶ Gürsad Küçük, Worldwide Customer/Partner Engagement Leader of Db2 and SAP Software, IBM Germany
- ▶ Carola Langwald, SAP on Db2 LUW Development Support, IBM Germany
- ▶ Thomas Matthä, Senior Development Engineer, SAP Certified Consultant, IBM Germany
- ▶ Beck Tang, SAP Integration and Support, Hybrid Data Management, IBM Analytics IBM Hybrid Cloud, IBM Canada
- ▶ Frank-Martin Haas, SAP SE
- ▶ Karen Kuck, SAP SE
- ▶ Jörg Schön, SAP SE

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 IBM Redbooks 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, IBM Redbooks
Dept. HYTD Mail Station P099
2455 South Road
Poughkeepsie, NY 12601-5400

Stay connected to IBM Redbooks

- ▶ Find 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>



Introduction

The information technology landscape is always changing. One of the changes that is prevalent in the industry is a growing interest in running workloads in the cloud. There are many benefits that can be achieved by moving to the cloud. Often, the benefits are financial in nature, but another consideration is that outsourcing the information technology mission allows more company resources to focus on their primary business objectives.

Moving workloads to another platform, whether it is a new cloud platform or not, involves migration of your data from your existing database environment to the new database. This book provides some tips and techniques to optimize that migration when using Db2 in an SAP environment.

The following topics are discussed in this chapter:

- ▶ 1.1, “Overview” on page 2
- ▶ 1.2, “SAP NetWeaver and Db2 in the cloud” on page 2
- ▶ 1.3, “Methods outside the scope of this document” on page 4
- ▶ 1.4, “Heterogeneous system copy” on page 4
- ▶ 1.5, “Unicode conversion” on page 5
- ▶ 1.6, “Database vendor change” on page 5

1.1 Overview

SAP offers software for various hardware platforms and database systems. The procedure to migrate an existing SAP system from one database system or operating system to another is known as a heterogeneous system copy or OS/DB migration.

In addition to the long-time existing platforms like IBM Power Servers with IBM AIX® as the operating system or servers that are running HP-UX, a new set of infrastructure platforms, the public cloud, is getting more attraction.

This is true not only for various cloud native applications by SAP, but also for classic SAP NetWeaver-based applications, such as Enterprise Resource Planning (ERP) and SAP Business Warehouse (BW).

SAP NetWeaver-based systems can be operated in the public cloud in an Infrastructure as a Service (IaaS) model.

SAP developed a set of tools with which customers can export their source database in a database-independent format and import it into the target database. These tools are integrated in the SAP Software Provisioning Manager (SWPM) as part of the Software Logistics Toolset (SL Toolset). With the same set of tools, the customer can convert a non-Unicode SAP system into a Unicode one and can decluster SAP table clusters.

This book describes various optimizations and best practices for converting a Db2 database to Unicode and for migrating an SAP system from a non-Db2 database to Db2. The focus is on Db2 specific configuration and optimization options.

1.2 SAP NetWeaver and Db2 in the cloud

There are many SAP solutions available in the cloud and natively built and designed for the cloud. SAP NetWeaver and those applications based on SAP NetWeaver, such as SAP ERP and SAP Business Suite, are not the typical cloud native solutions. However, they can run on hyperscaler platforms that are defined as Infrastructure as a Service.

There are three main models in cloud computing that do the management for you:

- ▶ Infrastructure-as-a-Service (IaaS)
- ▶ Platform-as-a-Service (PaaS)
- ▶ Software-as-a-Service (SaaS)

The models differ in which layers are managed by the cloud provider. The differentiating features for each model is shown in Figure 1-1 on page 3.

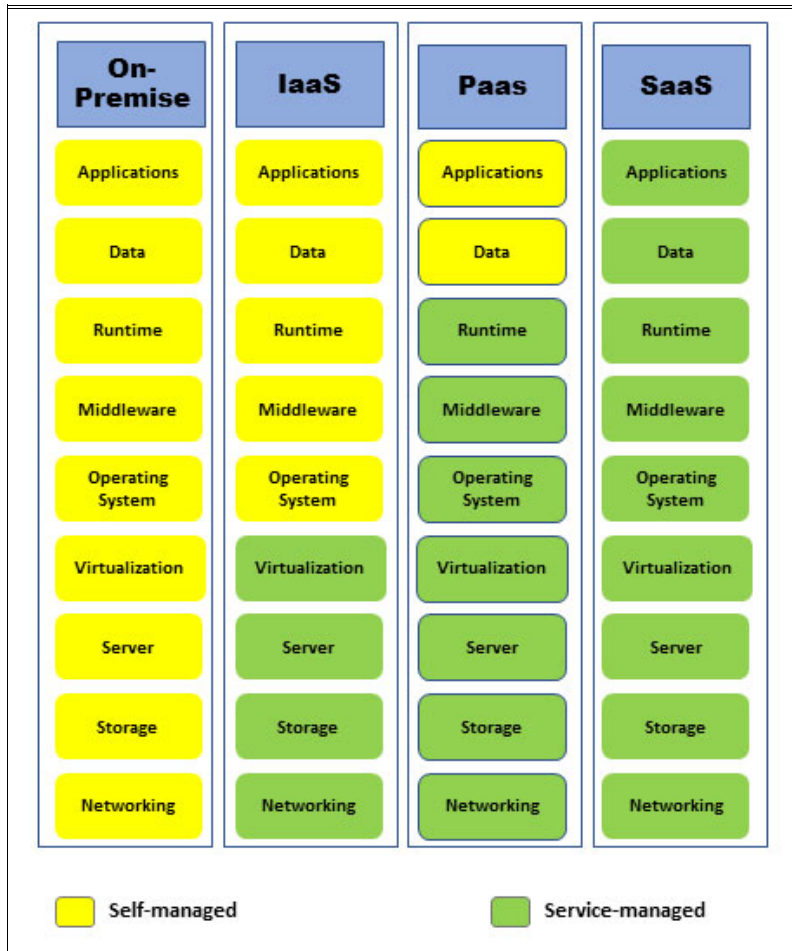


Figure 1-1 As-a-service models

IaaS

IaaS is defined as access to cloud-hosted computing infrastructure, which includes server, storage, and networking resources.

The user can provision the machines by using a graphical user interface (GUI) or by using API calls. The cloud service provider hosts, manages, and maintains the hardware and computing resources in its own data centers. IaaS is accessed using an internet connection, and customers pay for the resources through a subscription or on a pay-as-you-go basis. Typically, IaaS customers can choose between virtual machines or bare metal servers on dedicated physical hardware.

The benefits of this IaaS model are the increased flexibility to build and scale resources as needed. You can avoid up-front expenses, and you do not need to plan for a spike in the workloads. SAP Business Suite on Db2 is an example of an IaaS model.

PaaS

PaaS is another step further from on-premises infrastructure. In the PaaS model, a provider offers the services from the IaaS model and delivers this platform to the user as an integrated solution, solution stack, or service through an internet connection. The SAP Integration Suite, IBM Red Hat OpenShift or IBM Db2 Warehouse on Cloud are examples for a PaaS model.

SaaS

The SaaS model, also known as cloud application services, is the most comprehensive form of cloud computing service. It provides the services from IaaS and PaaS and enhances those models with services for the application and data. The model delivers an entire application that is managed by a provider, by using a web browser. The customer does not have to care about software updates, bug fixes, or maintenance for the whole stack. Some examples of SaaS applications are an email application with which you access your mails by using a web browser, SAP SuccessFactors, and Red Hat Insights.

Because IBM Db2 with SAP NetWeaver and SAP NetWeaver-based products are available in an IaaS model only, this book concentrates on the IaaS model. Also, all of the findings and recommendations in this book are also valid for virtualized on-premises environments and for bare metal server implementations.

1.3 Methods outside the scope of this document

There are various areas and tools that you can use to optimize the migration process. Documenting all of those tools is beyond the scope of this publication. Therefore, some commonly used tools are not covered in detail in this book, but they are mentioned here for your reference.

Readers of this publication should be familiar with the following tools and terms:

- ▶ SAP Software Provisioning Manager (SWPM)
- ▶ SAP Migration Monitor and SAP Distribution Monitor
- ▶ SAP Package Splitter
- ▶ SAP tables split tools like R3ta and SAPup

For more information about the preceding tools, see the following documentation:

- ▶ [System Copy for SAP Systems Based on the Application Server ABAP of SAP NetWeaver 7.3 EHP1 to 7.52 on UNIX](#). Target Databases: SAP ASE; SAP MaxDB; Oracle; IBM Db2 for z/OS®; IBM Db2 for Linux, UNIX, and Windows; MS SQL Server
- ▶ [System Copy for SAP Systems Based on the Application Server ABAP of SAP NetWeaver 7.3 EHP1 to 7.52 on Windows](#). Target Databases: SAP ASE; SAP MaxDB; Oracle; IBM Db2 for z/OS; IBM Db2 for Linux, UNIX, and Windows; MS SQL Server

In addition to the essential tools for a heterogeneous system copy, other options that are not discussed in detail and were *not* used while writing this book include the following examples:

- ▶ A feature called Database Migration Option (DMO) included in the SAP Update Manager
- ▶ The IBM Db2 family on platforms other than IBM Db2 for Linux, UNIX, and Windows

1.4 Heterogeneous system copy

The process of copying an SAP system to a different operating system or database platform is known as heterogeneous system copy. The process is illustrated in Figure 1-2 on page 5.

In brief, a heterogeneous system copy has two primary steps:

1. The database of the source system is exported by using SAP tools into a format that is both database and operating system independent.
2. A new SAP system is installed and the export from step 1 is used to load the database.

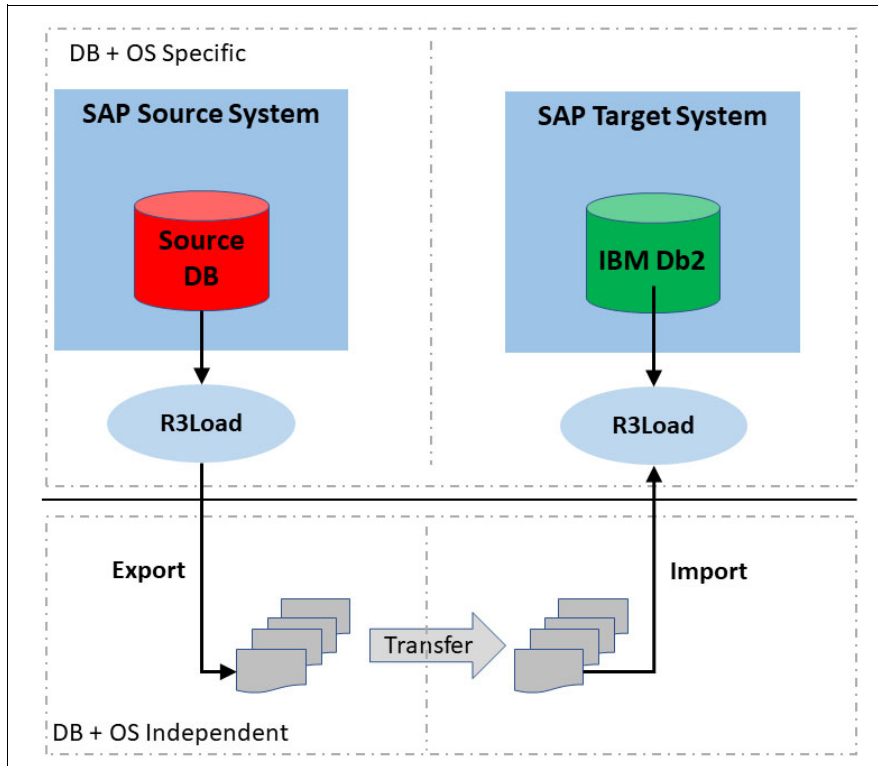


Figure 1-2 SAP heterogeneous system copy overview

While exporting or importing, the SAP system must be offline. No user activity is permitted. A typical amount of time that is scheduled to perform a heterogeneous system copy is one weekend.

In cases where the system is large or when the migration time might exceed the available downtime, you must apply special optimizations to the export and import process. This book describes those optimizations and preferred practices available for IBM Db2.

1.5 Unicode conversion

The technique of a Unicode conversion is similar to that of a database migration, which consists of an export and an import phase. They are both based on the use of the program R3load. The Unicode conversion itself is normally run during the export phase. Therefore, you can change the database for the target system without additional effort.

1.6 Database vendor change

The tools and techniques in this book can also be used to change the database management system from one vendor to another. If you are changing the database management system from any database platform that is supported by SAP NetWeaver-based applications to IBM Db2, most of the chapters in this book provide valuable information. For the export, you can consult documentation for the specific source database type and combine that information with the information about Db2 in this book.



Db2 with SAP in the Cloud

Running SAP NetWeaver with a Db2 database has been an option for many years in the on-premises environment. Several years ago SAP started to certify different cloud platforms to provide customers a choice on where to run their SAP environments.

This chapter provides the foundations that can be used to plan and size your environment for a cloud implementation by using Db2 as the database engine.

This chapter includes the following topics:

- ▶ 2.1, “SAP in the cloud” on page 24
- ▶ 2.2, “Db2 features for cloud deployments” on page 24
- ▶ 2.3, “Sizing of SAP systems for cloud service providers” on page 25
- ▶ 2.4, “Storage considerations” on page 26

2.1 SAP in the cloud

SAP NetWeaver on Db2 is available on multiple cloud platforms. The supported platforms are Microsoft Azure, Amazon Web Services (AWS), Google Cloud Platform (GCP), Alibaba Cloud, and IBM Cloud®. The availability of an SAP system as an IaaS model requires the cloud service provider to compile blueprint documents and to certify the virtual machines or bare metal servers with SAP.

For details about the prerequisites, supported virtual machines, and the types of bare-metal servers that are available with SAP, see the following SAP Notes:

- *2552731 - SAP Applications on Alibaba Cloud: Supported Products and IaaS VM Types*
- *1600156 - DB6: Support statement for Db2 on Amazon Web Services*
- *2233094 - DB6: SAP Applications on Azure Using IBM Db2 for Linux, UNIX, and Windows - Additional Information*
- *2927211 - SAP Applications on IBM Cloud Virtual Private Cloud (VPC) Infrastructure environment*
- *2414097 - SAP Applications on IBM Cloud Classic Infrastructure environment*
- *2855850 - SAP Applications on IBM Power Virtual Servers*
- *2456432 - SAP Applications on Google Cloud: Supported Products and Google Cloud machine types*
- *3000343 - SAP Applications on Google Cloud: Supported Products on Google Cloud Bare Metal Solutions*

Note: An SAP user ID is required to access the SAP Notes in the SAP support portal.

2.2 Db2 features for cloud deployments

Db2 offers different features that can be used with SAP applications. There are features that are mainly infrastructure-independent such as Db2 compression, Db2 native encryption, multidimensional clustering (MDC), and column-organized tables (IBM BLU acceleration®). All these features are fully supported independently if you are running on-premises or in the IaaS model. Also, the option to use massively parallel processing (MPP) also known as Database Partitioning Feature (DPF) is supported when running Db2 in the cloud.

A feature that is missing from this list of fully supported features is the usage of cluster managers for high availability solutions. The usage of cluster managers requires a documented blueprint by the cloud service provider.

For Microsoft Azure and Amazon Web Services, the cluster manager, Pacemaker, is documented and is the recommended solution. At the time of writing, for the Google Cloud Platform, this cluster manager is not yet certified.

All Db2 and SAP-supported cluster managers on the different cloud platforms are based on the Db2 High Availability Disaster Recovery (HADR) feature. HADR provides a high availability solution for both partial and complete site failures. HADR protects against data loss by replicating data changes from a source database, called a primary database, to the target databases, called the standby databases.

Cluster managers add the functionality of automated failover to the standby database during a planned or unplanned outage of the primary database server. The HADR feature can also be used for migration to the cloud infrastructure. Section 11.2, “Db2 HADR” on page 186 describes the option to run the primary server on-premises and build the standby database server in the cloud environment. This option reduces the required downtime.

The IBM Db2 pureScale® feature is a shared disk cluster with multiple active database servers and delivers high-availability and scalability. At the time of writing, IBM Db2 pureScale is not supported by any of the cloud vendors in combination with SAP applications.

2.3 Sizing of SAP systems for cloud service providers

The responsibility for sizing your SAP system in the cloud depends on the cloud model used. For SaaS model deployments, sizing is the responsibility of the cloud service provider because the full stack is provided by the cloud service provider. In the IaaS model, sizing is the responsibility of the customer because the customer must assign the appropriate virtual machines to an SAP landscape. Because only IaaS based implementations are supported to run SAP applications on Db2, the responsibility for sizing always falls on the customer.

When you move your SAP NetWeaver-based infrastructure to a cloud service provider, you have a series of VM instances or bare metal servers available that are certified for SAP. You are responsible for choosing which instance to use. The instances differ in CPU and memory configuration. In addition to the CPU and memory configuration, multiple types of disks are available for your storage configuration.

To determine the correct configuration that meets your system performance, an SAP sizing is required. Customers perform the sizing by themselves. For more information about SAP NetWeaver sizing, see [SAP Sizing](#). One tool that is often used to size the SAP landscape is the SAP Quick Sizer

As a result of this sizing process, you get an SAP Application Performance Standard (SAPS) number. The SAPS number for each of the certified virtual machines and bare-metal servers is documented in the SAP Notes that are mentioned in section 2.1, “SAP in the cloud” on page 24. Also included in those documents are the infrastructure characteristics of the servers, which include the number of CPUs and memory.

Table 2-1 provides some sample configurations derived from *SAP Note 2927211 - SAP Applications on IBM Cloud Virtual Private Cloud (VPC) Infrastructure environment*.

Table 2-1 SAP sizing examples

Server Type	vCPU	RAM	SAPS
mx2(d)-8x64	8	64	10.283
bx2(d)-2x8	2	8	2.306
ux2d-200x5600	200	5.600	215.570

With the results of the SAP sizing and the SAPS information in the SAP Notes for each cloud service provider, you can map the SAP landscapes to the correct virtual machines.

If you have an on-premises system on an operating system that is also supported by the cloud service provider, a good sizing method is based on the data collection from the existing system that will be moved to the cloud. Ensure that your cloud-based environment provides at least the same SAPS capacity as your existing system.

To gather this sizing data, you can use operating system tools such as iostat, vmstat, or the performance monitor on Windows operating systems. In addition, several 3rd-party tools exist for monitoring and collecting data. Because there is a large variety of tools available, a comprehensive list is beyond the scope of this book.

The data that is used in this book was gathered using the free of charge tool called Nigel's Performance Monitor (nmon), developed by Nigel Griffins, which is a standard tool that comes with the AIX operating system. For a Linux environment, it can be found and downloaded from [SourceForge](#).

The tool provides the ability to collect data in defined intervals over a period, and the collected data can be processed with Microsoft Excel to visualize the data graphically. See Figure 2-1.

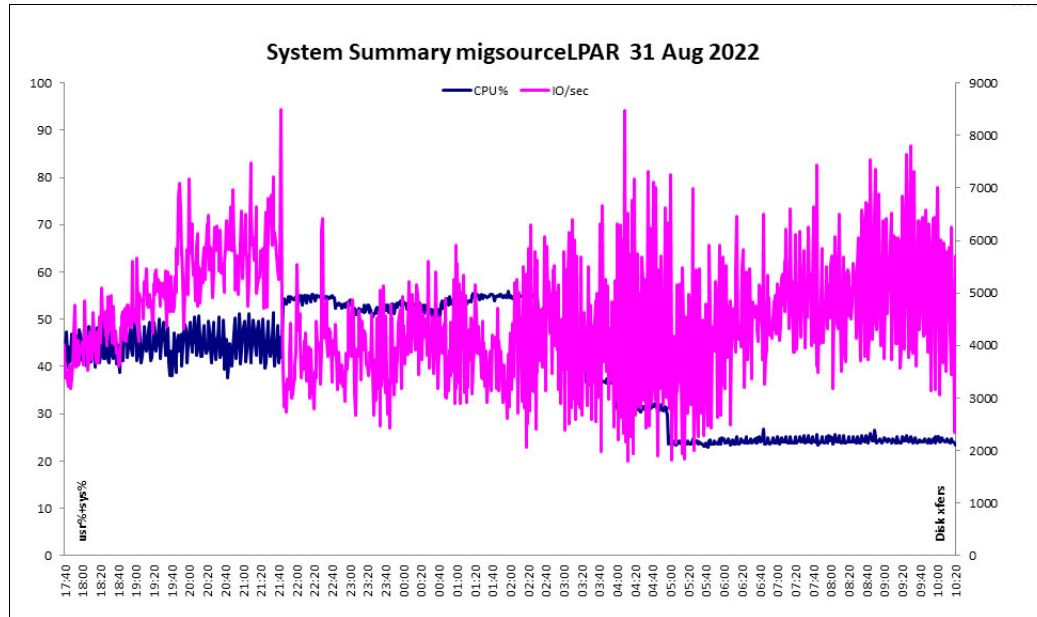


Figure 2-1 Nmon Graph example

When you collect the system resource data, be sure that the collected data includes periods of peak workloads. Collect and analyze the data during a high workload period, such as during a quarter-end closing process.

The nmon tool or other comparable tools do not provide SAPS numbers. However, after you gather the necessary CPU and memory requirements, you can map those to the certified cloud instances that are offered by your cloud vendor.

2.4 Storage considerations

The performance of the storage subsystem is also a key factor of the overall system performance. Therefore, pay special attention to the storage resources of your database server.

The cloud service providers offer different options for storage that range from classic hard-disk drives (HDD) to solid-state disk drives (SSD). The HDD offerings are entry-level resources that typically are not feasible for SAP NetWeaver database servers and can be used only for test or education systems.

Storage that is offered for SAP NetWeaver-based systems are usually SSD storage devices for the database server. Other storage types can be used for backup and for temporary or archived data.

Storage services offered by the cloud service providers have different names and characteristics. The following list provides examples of the different storage services that are typically used for SAP applications:

- Azure: Ultra Disks, Premium SSD, Standard SSD, or Standard HDD
- AWS: General Purpose SSD, Provisioned IOPS SSD, Throughput Optimized HDD, or Cold HDD.
- GCP: Persistent Disks with the types of Standard, Balanced, SSD, and Extreme.
- IBM Cloud: 3 different IOPS tiers or customized IOPS profiles.

The offerings for each cloud vendor also differ in characteristics. The following list provides examples of the features to consider:

- Availability options
- Replication capabilities
- Storage encryption
- Snapshot capabilities

Another important storage characteristic to be considered is read and write throughput for the disk. Throughput capabilities can be documented by using different units of measurement:

- Megabits per second (Mbps)
- Megabytes per second (MBps)
- Mebibytes per second (MiBps)

Carefully compare these details when comparing different cloud service providers.

The most important storage characteristic for Db2 and other relational database management systems is the number of input/output operations per second (IOPS). This applies especially to online transaction processing (OLTP) workloads such as SAP Business Suite.

Again, the different cloud service providers document the IOPS performance differently, both across vendors and sometimes within the same vendor for the different classes of storage, such as IOPS per gigabyte of capacity or IOPS per volume.

For example, if you provision a disk in the IBM Cloud you can choose between three, five or ten IOPS/GB. On Microsoft Azure, you can provision the Standard SSD with a maximum of 6,000 IOPS, depending on the size of your disk. Also, on Microsoft Azure you can provision Ultra Disks, which offer a maximum of 160,000 IOPS depending on the size of your disk.

IOPS performance often increases with the allocated capacity. Table 2-2 on page 28 shows an example of the correlation between capacity and IOPS. There is some difference in the details and ranges for the different cloud service providers, but the concept remains the same for each of the different cloud service providers.

Table 2-2 Example capacity versus IOPS

Size Range (GB)	IOPS range
10-99	100-1000
100-199	100-2000
200-499	100-4000
500-999	100-10000

During the sizing process, you might have a situation where you must allocate more storage capacity than required to store the data because of the IOPS requirements. For example, using the values in Table 2-2, if your database size is 700 GB with 20,000 IOPS required, then you must allocate two volumes to meet your IOPS requirements, though a single volume would provide enough disk space.

A good tool to help with the IOPS sizing is the history monitoring of your source system. If you monitor the IOPS over a longer period, you can assess the real number of IOPS required when you move to the cloud. Ideally, the IOPS for log volumes and data volumes are monitored and recorded separately. The benefit of IOPS monitoring is that the numbers are comparable and are independent of the operating system. If you plan to move from AIX on-premises to an IaaS model on Linux, the historical IOPS data is similar to the IOPS you generate in the cloud.



Migration essentials

This chapter discusses considerations for planning and performing a migration. The chapter includes recommendations for basic, standard, and advanced migrations and describes methods of optimizing resources during the migration to ensure that your migration is successful.

The following topics are discussed:

- ▶ 3.1, “Introduction” on page 30
- ▶ 3.2, “Top recommendations” on page 30
- ▶ 3.3, “Resource usage during the migration” on page 34
- ▶ 3.4, “Using the SAP migration monitor” on page 41

3.1 Introduction

The migration or heterogeneous system copy to the IaaS model requires a downtime of the system. This book includes features and options that help to minimize the system downtime during this process. Although various Db2 features can improve performance, the most impactful optimizations come from ensuring that you are effectively using the resources available to you through the optimal scheduling of the R3load processes.

The outcome of tests, which are combined with information gathered in consulting on customer migrations, is a series of tips and recommendations. The recommendations are separated into three areas:

1. A general set that is true for each heterogeneous system copy.
2. A set of recommendations for standard migrations.
3. Recommendations for downtime critical or advanced system copies that require tuning.

There is no clear definition of a standard or advanced migration. A typical scenario for an advanced migration might be a database with 10–20 TB that must be migrated with a technical downtime of less than 24 hours. However, if your database which is 600 GB must be copied in less than 3 hours, it might also require advanced techniques to meet those objectives.

In the presented cases, we distinguish between the following types of cases:

- ▶ Standard migrations that are run with SWPM and the integrated migration monitor only
- ▶ Downtime critical migrations where we recommend and assume that multiple instances of migration monitors are running and might not all be managed by the SAP SWPM

The SAP tools for heterogeneous system copies change over time and new functionality might be added. For the most recent information and recommendations, refer to SAP Note *3311643 - DB6: Recommendations for migrations or system copies*.

3.2 Top recommendations

The next sections detail the top 5 recommendations for each of the three categories that are specified in 3.1, “Introduction” on page 30.

3.2.1 General recommendations

Good scheduling and the correct parallelism concerning available resources are important factors to the overall result. The following recommendations apply to all migrations and can significantly impact the runtime:

1. Familiarize yourself with the SAP migration tools. Fully understand the capabilities and configuration options for R3load, migration monitor and how they interact with the SAP SWPM.

For more information, see Chapter 4, “Tools overview” on page 43.

2. Monitor not only the migration tools but also system resources like CPU, network, and IOPS. Understand the different steps R3load is running during export and import and how the steps contribute to CPU usage and I/O. This monitoring is essential to assess further optimization options.

For more information, see section 3.3, “Resource usage during the migration” on page 34.

3. Choose a suitable number for the concurrently running R3load import and export processes. A good estimate for both is 80%–100% of the number of CPUs on the export or import server.

For more information, see Chapter 5, “Export optimization techniques” on page 65 and section 6.6, “Import optimization” on page 106.

4. Perform one or more rehearsal migrations. Run the `migttime` package and analyze the bottlenecks of the rehearsal migration. A good source of monitoring information is the resource usage section at the end of each R3load log file.

For more information, see section 4.4, “Time analyzer (MIGTIME)” on page 47.

5. Run the migration check reports `SMIGR_CHECK_DB6` and `SMIGR_CREATE_DDL` before the migration and verify the correctness of the objects in the source database.

For more information, see Chapter 4, “Tools overview” on page 43 and Chapter 6, “Basic Db2 layout and configuration options” on page 85.

3.2.2 Standard recommendations

You can use standard recommendations if the migration monitor is started as part of the SAP SWPM migration with a single migration monitor instance running. The following standard recommendations can help achieve a certain throughput:

1. Implement a good storage layout and incorporate the best practices of your cloud service provider:
 - Use separate logging for data I/O and for temporary I/O.
 - Use multiple storage paths for data tablespaces, for example `sapdata1–20`, and try to achieve Db2 containers smaller than 100 GB.
 - Use multiple storage paths for temporary tablespaces, for example `saptmp1–4`.
 - Use SAP tablespace pools with a suggested 20 pools to a maximum of 60 pools but place large tables into separate tablespaces and introduce new data classes.
 - Provide enough log space. A good estimate is to use the planned configuration for production.

For more information, see section 6.1, “Planning your Db2 target system” on page 86.

2. Configure Db2 according to *SAP Note 2751102 - DB6: Db2 11.5 Standard Parameter Settings including Db2 Self-Tuning Memory Manager (STMM)*. Initially, calculate with one R3load process per vCPU or CPU and provide initial start values for STMM. You can calculate with at least 4 GB of memory per vCPU and therefore set the following values:

- `INSTANCE_MEMORY`

Set to 80% of the available physical memory in the database server in 4 KB pages.

- `DATABASE_MEMORY AUTOMATIC`

Set to automatic along with 80% of the `INSTANCE_MEMORY` as a starting value.

- `UTILITY_HEAP AUTOMATIC`

Use a starting value of 50000 4 KB pages * number of vCPU.

- SHEAPTHRES_SHR AUTOMATIC
Use a starting value of 500000 4 KB pages * number of vCPU.
- SORTHEAP AUTOMATC
Use a starting value of 250.000 4 KB pages.

For more information, see Chapter 6, “Basic Db2 layout and configuration options” on page 85.

3. After each rehearsal migration, use all self-tuning memory management (STMM) optimized parameter settings for the next rehearsal or for the final migration.

For more information, see section 6.14.1, “Using self-tuning memory management” on page 128.

4. Export the data unsorted whenever possible.

For more information, see section 5.2, “Unsorted versus sorted export” on page 66.

5. Use the following R3load options during the Import:

- COMPRESS_ALL
Use Db2 automatic dictionary creation to enable Db2 compression.
- ANY_ORDER
The migration throughput might show only a low single digit improvement, but it does not affect any migration negatively.
- -nolog
Use this option as it reduces logging I/O for small tables.

Important: Do not use the -nolog parameter of R3load when using parallel import with INSERT.

- DEF_CRT
Enable deferred table creation as the default option.
- -c 50000
A commit count of 50,000 rows is a good value if inserts on large tables occur. It is ignored if the db2load API is used.
- -stop_on_error
During a test migration, analyze and fix any errors.

This results in the recommended parameter string for R3load in the import_monitor_cmd.properties file:

```
loadArgs=-stop_on_error -loadprocedure fast
LOAD:COMPRESS_ALL:ANY_ORDER:DEF_CRT -nolog -c 50000
```

- Keep the default values for the environment variables:
 - DB6LOAD_CPU_PARALLELISM
 - DB6LOAD_DATA_BUFFER_SIZE
 - DB6LOAD_DISK_PARALLELISM
 - DB6LOAD_INDEXING_MODE
- Use R3load dump file compression when exporting tables with LOB fields. To enable the R3load dump file optimization for compression, use the R3load parameter **-compress adapt** during the export. The R3load parameter dynamically switches to

RLE based compression for cluster or INDX-like tables. As a result, the export throughput for some tables can increase significantly.

For more information, see Chapter 6, “Basic Db2 layout and configuration options” on page 85.

3.2.3 Five advanced recommendations

Although the set of standard recommendations can achieve a certain throughput for almost all heterogeneous system copies, the following 5 recommendations require additional effort for the IT environment or are more complex to set up and orchestrate. Consider them only if your downtime window requires additional optimization:

1. Use a dedicated application server for the export. Exporting the system uses a large amount of CPU resources. The workload is not only database related but can also include CPU usage for code page conversion, compression of export dump files, and declustering. If you decide to use dedicated application servers for the export and are using Db2 11.5.7.0 SAP5 or later, enable the Db2 **query block prefetch** feature.

For more information, see [New Query Block Prefetch Feature with Db2 11.5.6](#). For more information, see section 5.5, “Local or remote export server scenarios” on page 76.

2. Preallocate the tablespaces. Analyze the tablespace sizes after a first test migration and then create the tablespaces manually with the initial size of the tablespaces. You can combine this feature with setting the Db2 registry variable `DB2_USE_FAST_PREALLOCATION = OFF` to minimize fragmentation on the file system level as much as possible.

For more information, see section 6.4.8, “Preallocation of tablespaces” on page 93.

3. Use the environment variable `DB6LOAD_CPU_PARALLELISM` to optimize the import of selected large tables. The default for CPU parallelism of 4 for R3load is a good balance between overall performance and resources usage. In a dedicated migration monitor environment, you can set this environment variable to a larger number of 8–12 for the largest tables. The import might improve by 10% for those tables, whereas most of the tables are imported with a balanced configuration. Using it for all tables might result in a similar performance increase but might also increase CPU usage.

For more information, see section 6.7.2, “`DB6LOAD_CPU_PARALLELISM`” on page 111.

4. Optimize Db2 statistics collection. SWPM does not enable Db2 automatic RUNSTATS for the database until the import is completed. One method of optimization is to enable the automatic RUNSTATS feature when the import starts. This method might add a few additional CPU cycles and I/O operations on the target, but it can be beneficial. Use the advanced statistics collection feature with the **db2look** utility only if it is required and for only a selected set of tables.

For more information, see section 6.12, “Optimizing statistics collection” on page 123.

5. Optimize any table splits. If you are using table splitting, use a reasonable number of splits for the export and import in the range of 10–20 splits per table or a maximum of 100 GB of data volume per table split. Also, set an appropriate number of concurrent R3load processes in the `orderby.txt` file. The total number of concurrent R3load processes for the split tables and the remaining packages should not exceed the number of available CPU or vCPU capacity. Use the R3load parameter “`SPLITTED_LOAD`” to import the tables together with the environment variable `DB6LOAD_TEMP_TBSPACE` and without the R3load option `DEF_CRT`.

It is recommended that you read section 7.1, “Socket transfer and table split overview” on page 134 carefully and assess the feasibility of the R3load Option “`LOAD_FORCED`” and the usage of parallel INSERT.

3.2.4 More recommendations

This section includes other important recommendations to consider.

After our work with the various tuning options for heterogeneous system copies, we have found options that are either specific to a certain aspect of the process or have a significant impact on the process flow. Some of them can provide significant improvements. Before you implement any of these recommendations, read the full details about these optimizations so that you can fully understand the boundary conditions and side effects:

- ▶ Compression optimization. The R3load option COMPRESS_ALL can provide good compression rates with the best import throughput. If you meet your downtime window and want to optimize the size of your target database, you can use the R3load options OPT_COMPRESS or FULL_COMPRESS. Both options can significantly increase the import runtime. If you decide to use the option OPT_COMPRESS, consider also using the environment variable DB6LOAD_MAX_SAMPLE_SIZE. You can limit the number of tables for those options by using the orderby.txt file or a dedicated migration monitor instance.
- ▶ The Db2 registry variable DB2_SMP_INDEX_CREATE can decrease the time for creating the indexes on a table but can significantly increase the CPU and I/O workload. The registry variable must be set on the Db2 instance level and is therefore active for all processes. You can use this variable and set it to a large number, for example 64, after most the R3load processes are completed. Use it only for the indexes of the largest tables if you have available CPU and I/O resources.
- ▶ If you are migrating an SAP Business Warehouse system with column-organized tables, such as IBM Db2 with BLU Acceleration, use Db2 11.5.8 or later because the throughput with the db2load API increases by 30%–40% with this Db2 version. Plan for a utility heap of at least 1,000,000 pages instead of 50,000 pages per concurrent R3load process. Also, plan for increased CPU usage, which results in fewer concurrent R3load processes. For each concurrent R3load into a column-organized table, ensure that there is 128 GB of free space in the instance home directory for the process to create the compression dictionary.
- ▶ You can use a larger extent size for tables containing large tables. Results show a single-digit increase in throughput with an extent size of 16. You can create tablespaces with an extent size of 16 for large tables that are in a dedicated tablespace or for dedicated tablespaces that serve the temporary tables during a *splitted load*.

Important: Do not confuse extent size with page size. Do not use a different page size for the tablespace.

- ▶ The R3load socket option is an alternative to table splitting because it does not write the compression dump files to disk and therefore saves I/O and CPU resources for dump file compression. Consider this option for tables with LOB fields that do not compress well to mitigate the effect of increased network bandwidth usage.

3.3 Resource usage during the migration

A dedicated chapter about monitoring is included in this book. See Chapter 10, “Tips for monitoring” on page 175. This section provides an overview and explains the different types of resources involved. If you understand the correlation of certain steps in the process to resource usage, it can help you find the optimal setup for your heterogeneous system copy.

The main resource for a migration is the available compute power, which is determined by the number and performance of the available central processing units (CPUs), the available

memory, and the throughput of disk read/write operations. Another important resource is the network connection and its capability to transport data from the source to the target system.

The available resources must be used in the best possible way. Optimal resource usage includes the following factors:

- ▶ Use but do not overuse your resources.
- ▶ Ensure the constant use of resources.
- ▶ Balance the resources, such as CPU, I/O, memory, and network.
- ▶ Optimize the process to achieve the best overall throughput and do not perform local optimizations.
- ▶ Avoid unnecessary steps, tasks, and resource consumption.
- ▶ Use resources that are beneficial for a certain step in the migration.

3.3.1 Using your resources.

During the migration, use all the available resources, but avoid an overcommitment of resources, which can increase the overall runtime.

In Figure 3-1, a disk is 100% busy.

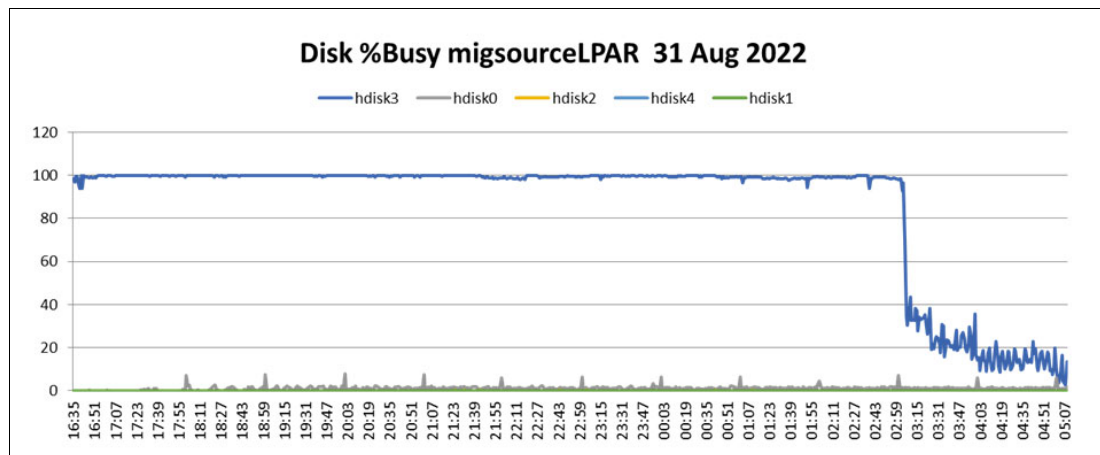


Figure 3-1 Disk Usage

This indicates that the storage subsystem is heavily loaded and there is no headroom for peak workloads. If the disk is constantly busy at 100%, disk queuing might increase, which can lead to increased load on the system because of increased wait times. The increased load on the system can slowdown overall performance.

The same is true if the CPU or network bandwidth are constantly at or near 100% usage. Figure 3-2 on page 36 shows CPU resources being used 100% over a long period. The graph also shows that the number of blocked processes is high between 08:30 and 10:00. Considering these facts together indicates an overloaded system. Decreasing the workload can usually help to improve overall performance.

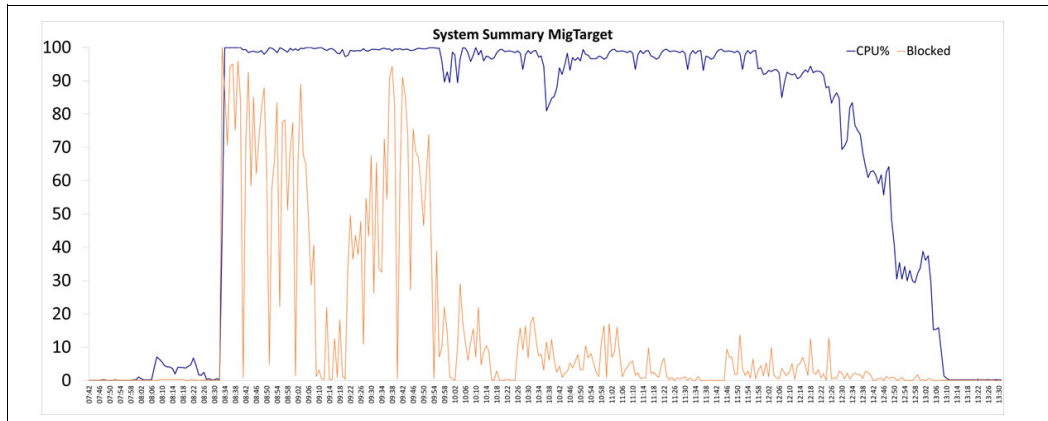


Figure 3-2 CPU usage

The information in Figure 3-2 is a good example of a system that is overloaded by too many parallel R3load processes. A good rule is not to start more R3load import processes than 80%–100% of the number of CPUs available.

3.3.2 Ensuring the constant usage of resources.

Using the resources in an optimal way during the whole export and import process delivers the best result. However, optimizing the process is one of the challenging parts of the migration, and you might not always be able to achieve an ideal resource usage. A good set of tools is the SAP Time Analyzer reports and in particular, the Package Time Diagram as an HTML file. Figure 3-3 shows an example report that provides the runtime and schedule of the exports or import processes. You can use this report to optimize the schedule of the packages. By comparing these reports with resource usage graphs of CPU, storage, and network, you can identify resource bottlenecks.

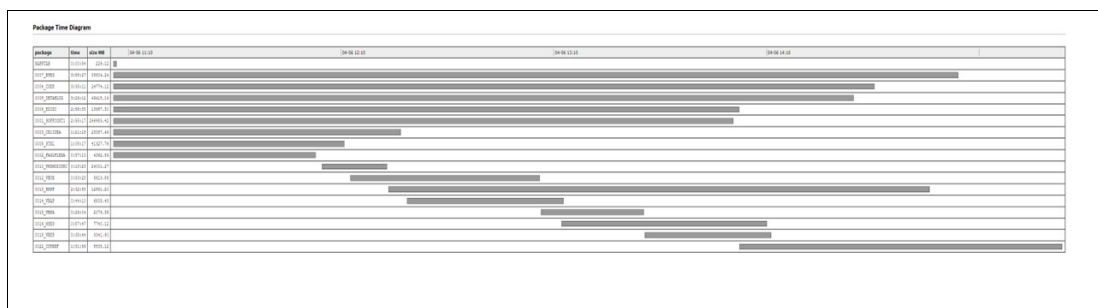


Figure 3-3 Package Time Diagram

3.3.3 Balancing the resources

One resource can create a bottleneck although other resources are still available. Experiencing a resource bottleneck does not mean that there is no further improvement possible. For example, the step of creating indexes for tables is an I/O intensive procedure. So, if you are experiencing a network or CPU bottleneck but have I/O resources available, try to schedule the R3load processes in a way that the index creation phase happens during that period. This requires a good knowledge of the SAP Migration Monitor, and often it is not easy to accomplish. Another more obvious example of a potential bottleneck is if the CPU resource usage reaches its limits, but you have network bandwidth available. In this case, it might be a good option to run the export or import on an additional machine. A typical scenario might be

a migration from an older generation of machine on-premises with an import to a more modern machine in the cloud. In this case, it might be beneficial to use the target machine to also perform parts of the export. Schedule the export in such a way that it does not overload the network bandwidth.

3.3.4 Achieving the best overall throughput.

Typically, either one table or a few tables determine the overall runtime of the migration, so the optimization efforts can center solely on a large table. Often the solution is to look at other tables or packages and to reduce their resource usage. By doing so, more resources can be available for the table that determines the overall downtime.

As an example, during one migration, a large z-table determined the overall migration time with a time of 27 hours.

Figure 3-4 shows the import times of the table EDI40 of the same migration that was split into 60 parts. The import runtime varies between 2–4 hours. This means that the import runtime varies by a factor of 2 although the imports were almost identical in the number of records.

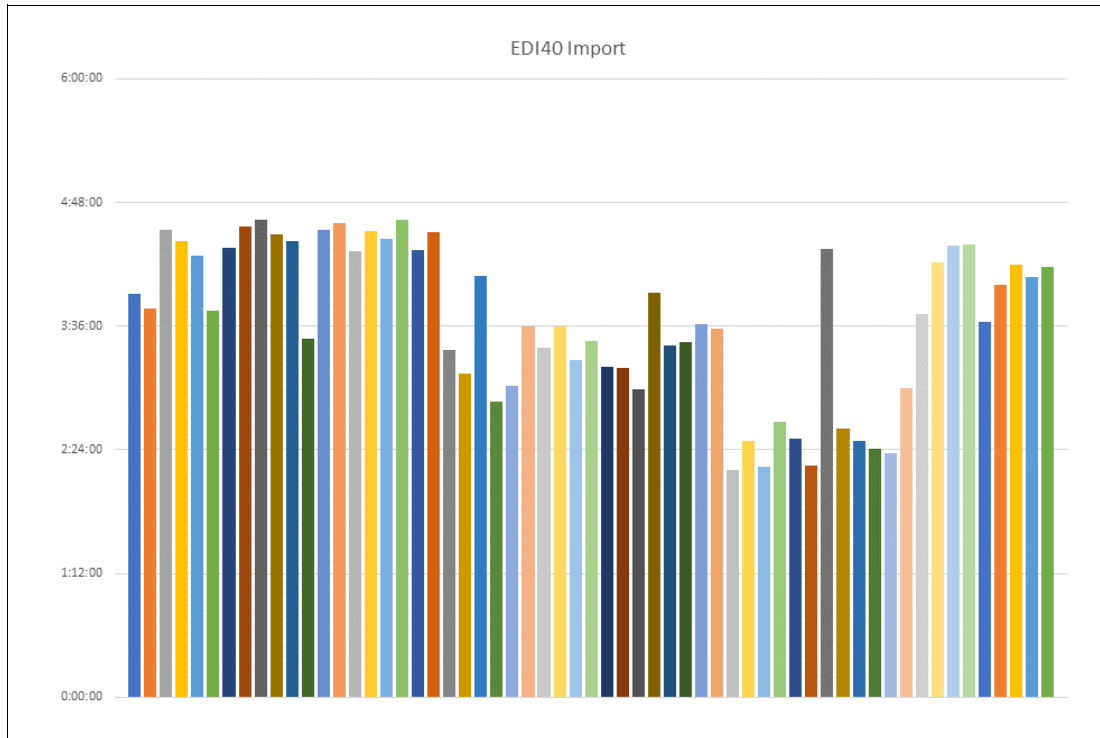


Figure 3-4 EDI40 Import runtimes

Further analysis shows that the system had limited CPU availability during a large portion of the time, and the import for the large z-table ran alone for several hours. To resolve this issue, some parallel R3load processes were reduced to make CPU resources available for the z-table import. The import times for the EDI40 table increased, but the large z-table import was expedited. As a result, the downtime target was met because the duration of the full migration was reduced from 27 hours to 24 hours.

3.3.5 Avoiding unnecessary steps, tasks, and resource consumption.

Avoiding unnecessary work is an obvious improvement to many processes and includes heterogeneous SAP system copies. Therefore, a large part of this book explains which unnecessary steps you can avoid.

Some of the most relevant examples are mentioned in this section, but you can find more details in other sections in this book.

If you use the R3load Option `LOAD_FORCED` together with a split export, Db2 loads each of the parts sequentially into the target database. If you choose to create the indexes on the table before the data is loaded, the complete index is rebuilt after each part of the table is loaded. Db2 eventually switches from this rebuild to an incremental mode based on internal calculations. However, a rebuild happens several times. To avoid this, either create the indexes after the LOAD completes or use the incremental indexing mode.

Another example are the flushes of the buffer pool after each LOAD operation completes. You cannot prevent them from happening, but you can decrease their negative impact by allocating a smaller buffer pool. This leads to a much smaller number of pages to be flushed and is therefore more effective.

3.3.6 Using the most suitable resources

The best way to fulfill the migration tasks is by using the correct resources. The largest part of this publication is about optimizing resource usage. To achieve this correct resource usage, a good knowledge of Db2 and R3load is required.

Table 3-1 provides some guidance about the resources that are used by the various steps of the migration process.

Table 3-1 Resource usage

	CPU	Disk	Memory	Network
Database SQL query	x	x	x	
Data compression and decompression	x			
Code page conversion	x			
Data fetch	x			x
Endianness conversion	x			
Unicode conversion	x			
Compression and decompression of export dump files	x			
Read/Write of dump files		x		x
Load or insert data	x	x		
Index create	x	x	x	
Build tables statistics	x	x		

During the R3load export, the following steps are run:

- ▶ Query on the source database tables with an SQL statement to export the data
This step typically involves CPU, disk, and memory resources.
- ▶ Optional data decompression
The source database might use data compression features, so data must be uncompressed when exported with R3load.
- ▶ Data fetch
This process transfers the data from the database to the R3load application. The networking resource usage depends on where you run the R3load export. If you run the R3load export on a dedicated application server or the target database server, data must be transferred over the network.
- ▶ Code page conversion from UTF-8 to UTF-16
Db2 stores data in UTF-8 format and SAP applications use UTF-16. This code page conversion can be run quickly, but requires a small amount of CPU resources.
- ▶ Optional endianness conversion
This conversion step is required if you move from a system with big-endian processor architecture like IBM POWER® to X86-based Linux systems with little-endian processor architecture. During R3load processing, you can switch the endianness by specifying the appropriate code pages, 4102 or 4103. This endianness conversion can be done during the export or the import.
- ▶ Optional declustering and depooling of data
During the export, you can decluster table clusters or depool pool tables. For more information, see SAP Note 2227432.
- ▶ Optional Unicode conversion
If your system is still using non-Unicode, you can convert to Unicode during the export. This step cannot be run during the import.
- ▶ R3load compresses the dump files to save space for the export
The compression rates depend on the table content and structure. Typically the export dump is only 10% of the data table size in the database.
- ▶ R3load writes the data into the export dump files.

When the export of a table or package is completed, the data is moved or copied to the target system. If the migration is to the cloud, the data transfer can be through a shared file system or through the use of data transfer tools. This step usually impacts network resources and to some extent the storage resources as well.

When the data is imported, the following major steps are run:

- ▶ R3load reads the data from the export dump location.
- ▶ The data is decompressed.
- ▶ Optionally convert the endianness.
This conversion step is required if you move from a system with big-endian processor architecture like IBM POWER to X86-based Linux systems with little-endian processor architecture. During R3load processing, you can switch the endianness by specifying the appropriate code pages (4102 or 4103). This endianness conversion can be done during the export or the import.
- ▶ Code page conversion from UTF 16 to UTF-8.

Db2 stores data in UTF-8 format and SAP applications use UTF-16. This code page conversion can be performed quickly but requires a small amount of CPU resources

- ▶ LOAD or INSERT the data into the database table.
- ▶ Optionally compress the data.

As part of the target database feature usage, data is compressed on disk. To do so, a compression dictionary is built on table and page level.

- ▶ Create indexes.

Other than the raw data, the target database consists of at least one and often multiple indexes that must be created.

- ▶ The Db2 cost-based optimizer requires metadata and statistical information about the tables.

This data is compiled by the RUNSTATS component of Db2.

An important fact about the usage of R3load is that the process is single-threaded. Therefore, some CPU resource-intensive steps cannot easily be scaled. An example is the compression and decompression of the export dump files. The only option to scale those steps is to split the packages and tables into multiple R3load processes. For many of the database-related steps, Db2 can use symmetric-multi-threading (SMT) and parallelization within the database management system. One example is the use of Db2 intra-partition parallelism during the export or index creation.

The following list provides additional examples:

- ▶ Memory optimization

Some of the most prominent examples of optimal resource usage are the efficient use of memory. As discussed previously, a smaller buffer pool can help to avoid the impact of buffer pool flushes. Furthermore, a smaller buffer pool uses less memory that can then be used for sort operations. It makes a significant difference if you sort in memory or spill intermediate objects to disk.

- ▶ Optimal CPU usage

optimal in the sense of relation of CPU usage to throughput): A Db2 load can use all the CPUs on the host. Using a CPU parallelism of 16 can ensure better throughput than a CPU parallelism of four. However, the benefit is typically only in the range of a 3-5% improvement. Therefore, by default R3load uses a CPU parallelism of four to achieve the optimal performance with a CPU resource usage.

- ▶ Efficient use of network resources

If you decide to use a dedicated host for exporting data, consider using the Db2 query prefetch feature. By doing so, you avoid unnecessary time that is spent waiting for the database to send data. However, the optimized data transfer between the Db2 server and the remote R3load might reduce the bandwidth available between your on-premises source and the migration target. Therefore, consider using additional local servers for the export instead of the target machine.

There are many more examples of optimal resource usage in the next chapters of this publication.

3.4 Using the SAP migration monitor

If you know at what point your resources are in high demand and if you know the steps of R3load that contribute to the resource usage, you can use the SAP Migration Monitor (MigMon) to schedule the R3load processing in a way that it uses the resources optimally.

As a first approach, you can use the migration monitor that is started by the SAP Software Provisioning Manager (SWPM) as part of the SAP SL Toolset 1.0.

One of the most powerful options of the migration monitor is the `orderBy` configuration option. You can use it to schedule different packages or tables with a different number of R3load Jobs. This configuration option is often used for table splits during export and import. Also, you can also use different load arguments for different packages.

3.4.1 Configuring monitors for optimization

Different monitor configurations can help with optimization of resources.

Using the `orderBy` option

You can use the `orderBy` configuration option to achieve different optimization goals:

- ▶ Start a package or table before other tables with different options.
For example, if you use INSERT for a table containing large LOB fields instead of the LOAD feature of Db2, you can avoid interference with the other processes that are caused by logging.
- ▶ Control the number of processes per package or table.
This configuration setting is often used for table splits. The number of total processes is the sum of the jobs defined in the `import_monitor_cmd.properties` file and the configuration in the `orderBy` file.
- ▶ Specify different arguments for export or import.

Using the `import_monitor_cmd.properties` file

The following example shows the usage of different load arguments for a specific table. The `import_monitor_cmd.properties` file specifies the default load arguments for the process. This is shown in Example 3-1.

Note: Example 3-1 and Example 3-2 do not show a complete configuration but show only the parts that are required to illustrate the use case.

Example 3-1 Snippet of `import_monitor_cmd.properties`

```
jobNum=20

loadArgs=-stop_on_error -loadprocedure fast LOAD:OPT_COMPRESS_ALL:ANY_ORDER -nolog
-c 50000

orderBy=/export/orderby.txt
```

In the configuration shown in Example 3-1, the migration monitor uses 20 parallel R3load processes to import packages and the Db2 compression feature `OPT_COMPRESS` to load the data into the target database. In addition, the migration monitor uses the information in the file `/export/orderby.txt` to import the data in a specific order.

The `orderby.txt` file that is shown in Example 3-2 contains the information for the table `SOFFCONT1`.

Example 3-2 Snippet of `orderby.txt`

```
[SOFFCONT1]

jobNum=1

loadArgs=-stop_on_error -loadprocedure fast LOAD:COMPRESS_ALL:ANY_ORDER -nolog -c
50000
```

This configuration option sets the number of parallel `R3loadjobs` for this table to 1 and does not use the `OPT_COMPRESS` feature for this table. By using this option, the table might not be fully compressed on the target, but the additional time for the 2-step compression is eliminated.

Using the `orderby` file to configure and schedule the `R3load` processes is often enough to achieve a decent migration throughput. Sometimes, it can be beneficial to run multiple instances of the migration monitor. To do so, you must setup the multiple instances manually and configure them.

3.4.2 Using multiple monitor instances

You can use multiple migration monitor instances to achieve different optimization goals.

Starting a package or table first

The migration process starts when the specific instance of the migration monitor is started. A scheduling starts the instance of the migration monitor manually. For example, it is possible to start the migration process for all small tables after the large tables are completed.

Controlling the number of processes per package or table

You can control the number of jobs by using the `orderby` option as well. When you use multiple migration monitor instances, the number of jobs that are configured in the `import_monitor_cmd.properties` file can be configured dynamically and changed during the migration process, but it is static for the `orderby` option. This can be useful for the first set of test migrations to determine the best number of processes.

Using special optimization features

You can specify different load arguments for a set of packages or tables with the `orderby` functionality. However, other features cannot be set for specific tables or packages. For example, consider the configuration option `DB6LOAD_CPU_PARALLELISM` that is set using an environment variable on the operating system. This option determines the number of threads to be used by the Db2 `LOAD` utility.

The SAP default setting is to use a parallelism of 4 as it is a good balance between performance and resource usage. You can increase the parallelism for all packages or tables by setting the environment variable `DB6LOAD_CPU_PARALLELISM` globally in the user environment. However, it might be a better use of resources to run most tables with the SAP default and to change the parallelism only for a selected set of tables. A possible scenario for this option is to use it only for the table that most affects the overall migration downtime.

Resources usage distribution

Your resource monitoring might show high usage of CPU resources on the export database server and that a large portion of the CPU usage is due to compression of the R3load dump files. You can reduce CPU usage on the export server by running R3load on the target server so that the target server reads and compresses the exported files.

Assess if the potential increase in network usage, and the competition for resources between the export process and the import process provides a benefit to the overall migration. An enhancement of this concept is to use dedicated application servers to either run import or export processes. The application servers are available on a temporary basis and the SAP applications are never started on the application server. Run only the migration monitor and the R3load processes on this server. If you are using a cloud infrastructure, the flexibility of that cloud environment provides the ability to create these temporary servers.

If the original database server hardware is older, it might be beneficial to use a server for the export that is different than the database server because the single thread performance of older CPUs is not as good when compared to newer machines. Also, use a different server if the number of CPUs on the export database server is limited. By using a different server, you can offload the R3load specific processing of code page conversion or dump file compression, and leave only the database workload running on the original database server.

Figure 3-5 shows a possible setup of multiple migration monitor Instances.

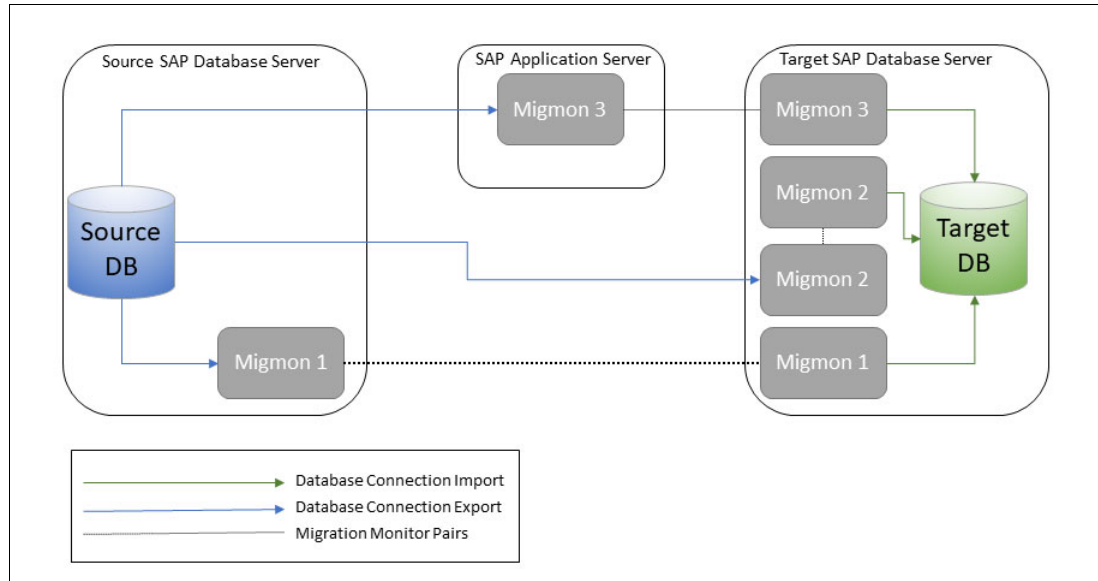


Figure 3-5 Overview of multiple migration monitor Instances



Tools overview

The following chapter provides an overview of the available SAP system copy tools. You can use these tools for both homogeneous and heterogeneous system copies. Tools that are used for heterogeneous system copies are often also referred to as migration tools.

This chapter is not intended to replace the SAP training or other available documentation. The chapter introduces the tools that are required for the Db2 optimizations and provides some pointers and hints that were found during testing.

This chapter includes the following topics:

- ▶ 4.1, “SAP Software Provisioning Manager” on page 44
- ▶ 4.2, “Embedded load tools” on page 44
- ▶ 4.3, “Migration monitor” on page 46
- ▶ 4.4, “Time analyzer (MIGTIME)” on page 47
- ▶ 4.5, “Tools for package and table splitting” on page 53
- ▶ 4.6, “SAP Migration Checker” on page 54
- ▶ 4.7, “SAP data definition tools” on page 56

4.1 SAP Software Provisioning Manager

The installation tool SAP Software Provisioning Manager (SWPM) is part of the SAP Software Logistics Toolset and is used to install SAP systems or to unload or load systems during a system copy procedure. SWPM invokes other tools, for example R3ldctl, R3szchk, R3load, and SAPuptool. SWPM controls the entire installation and migration process by providing the following features:

- ▶ Creating users or groups at the operating system level
- ▶ Adapting file system permissions
- ▶ Installing SAP binary files (SAP kernel)
- ▶ Triggering the database unload and load process
- ▶ Triggering post processing, such as database statistics, depooling and declustering of tables, and other processes

To simplify the migration process, the SAP installation tool provides software lifecycle options. For example, the following options are available for a source system during a heterogeneous system migration:

- ▶ Export preparation
- ▶ Table splitting preparation
- ▶ Database instance export

4.2 Embedded load tools

Since the release of SAP SL Toolset 1.0 SP21, the tools R3load, R3ldctl, R3szchk, and SAPuptool are bundled and embedded into the SWPM. This bundle is called LOADTOOLS.SAR, which exists for unicode and non-unicode.

SWPM does *not* use the installed kernel binary files. Instead, SWPM uses the embedded LOADTOOLS to guarantee matching versions during the entire system copy process. Also, a new replacement for the R3ta utility was introduced which is called SAPuptool. This tool is used during table splits for SAP release version 7.4 or later because it is faster at calculating where a table splits.

Note: LOADTOOLS support starts with SAP release 7.4 and is not used in SAP release 7.31 or earlier.

If you must manually update LOADTOOLS, check the following SAP notes for the most recent information:

- [2472835](#) - *Embedding load tools in Software Upgrade and SWPM*
- [2557361](#) - *How to update R3load, R3szchk and R3ldctl for SUM and SWPM*

The following sections describe the three main tools in the SAP SL Toolset: R3load, R3szchk and R3ldctl.

4.2.1 R3load

R3load is the core migration tool. It exports SAP ABAP table data from the source database and imports it into the target database. In addition to this base function, it offers advanced options. For example, it supports Db2 adaptive compression while importing data. To ensure

the functionality, R3load reads and writes several files that contain metadata. The functions provided by R3load and its metadata files are shown in Figure 4-1.

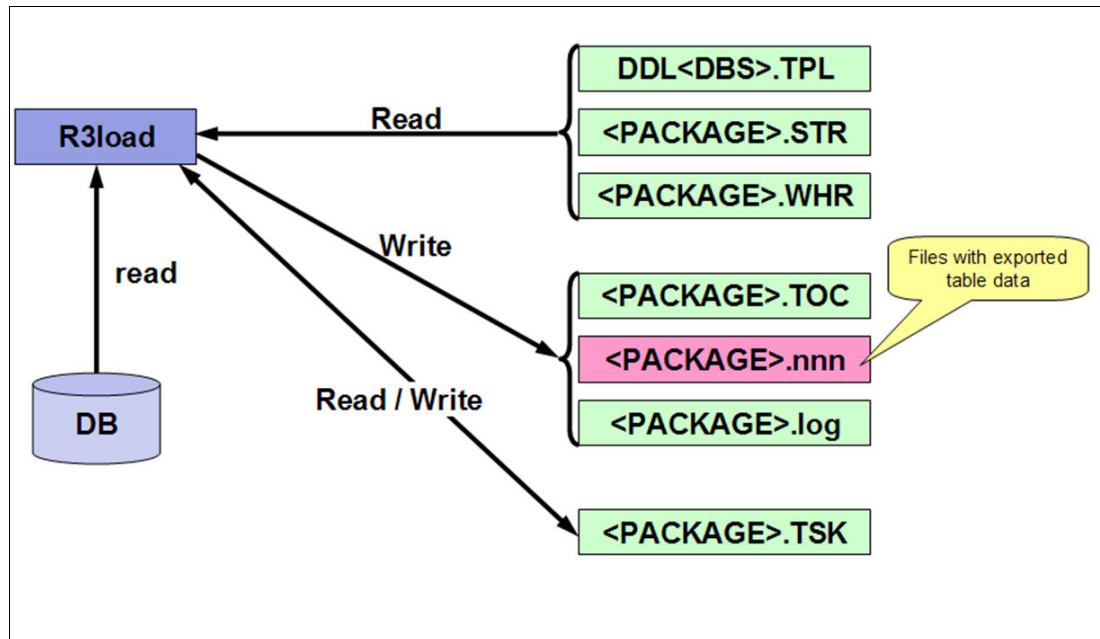


Figure 4-1 Tasks Performed by R3load During the Export

4.2.2 R3ldctl

This tool unloads SAP ABAP data dictionary structures from the source system. It generates structure files (*.STR files) that describe the definition of tables, indexes, and views. In addition, it generates more files that are required during the migration:

- ▶ Database-specific template files (*.TPL files) with definitions of DDL statements, such as statements to create a table or to drop a table.
- ▶ If there is declustering and depooling during the export, the R3ldctl generates logical files during the preparation, so the import can create the new structure during the import.
- ▶ The files that end with .WHR can include SQL statements to export only chunks of a table and are described in detail in section 7.3, “Table splitting” on page 139.
- ▶ The task files (*.TSK) control the tasks of the utility and log the information about completed or failed tasks.
- ▶ The table of contents file (*.TOC) stores the information that is used by R3load to find the data offset and length of the data portion in a dump file.

Important: During production migrations, the R3ldctl tool must be run while the SAP application is offline and isolated to avoid potential loss of data.

4.2.3 R3szchk

R3szchk computes the size of tables and indexes for the target database.

The functions provided by R3ldctl and R3szchk are shown in Figure 4-2.

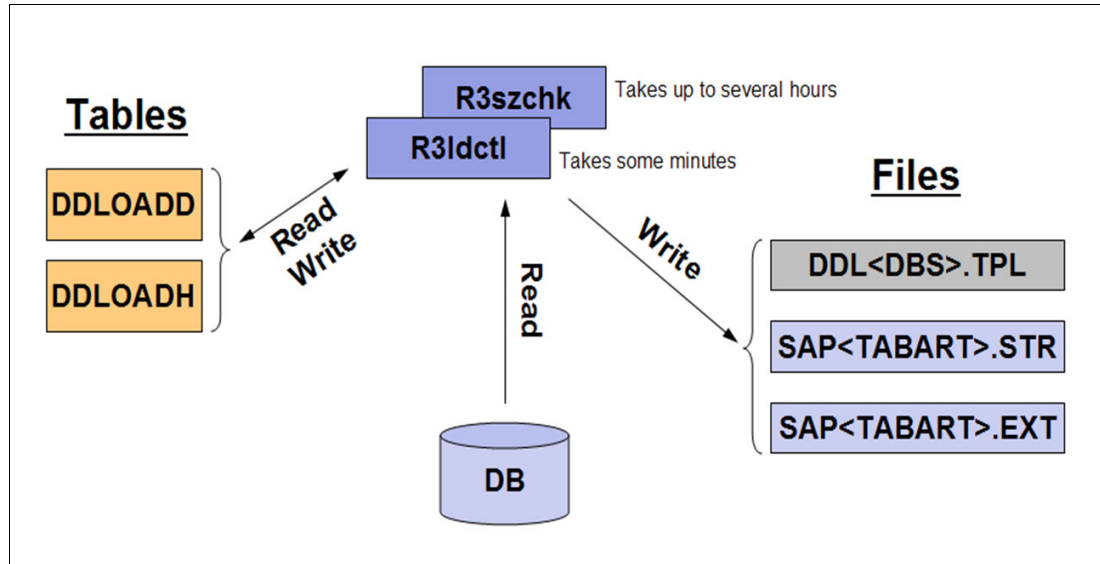


Figure 4-2 Tasks Performed by R3ldctl and R3szchk

R3szchk might run for a long time sometimes. The way to calculate the target database size uses `select count(*)` statements against all tables. This is true if the database management system is changed during the migration. If you experience long runtimes of R3szchk, check if R3szchk is using the `select count(*)` statement for the tables. You can use db2top or other database monitoring tools to validate In-Flight Statements.

Note: R3szchk requires the STR files created by R3ldctl to calculate the target size. If you run R3szchk and R3ldctl while the system is running, you must not make any data definition changes, by new transports, for example. These changes are not reflected in the STR Files, so the export is not consistent or fails.

4.3 Migration monitor

The migration monitor (MigMon) is a tool to schedule, run, and monitor the export and import processes. It orchestrates executable programs like R3load and R3ldctl.

The following list describes the main aspects and attributes of MigMon:

- ▶ Allows advanced control of R3load export and import
- ▶ Automates transferring the dump files from the source to the target system by using FTP
- ▶ Supports parallel unload / load processing by using an NFS share
- ▶ Controlled by properties files
- ▶ Rereads property files at regular intervals so some attributes like the number of processes can be adjusted dynamically

The tool is fully integrated into the system copy procedure of the Software Provisioning Manager as part of the SAP SL Toolset 1.0. Although it is part of the system copy tools, the migration monitor is an important component. It is a good idea if you are familiar with how to use the tool manually and when it is part of the Software Provisioning Manager.

The tool is described in *SAP Note 784118 - System Copy Tools for ABAP Systems*. Its parameters, functions, and control files are explained in more detail in the *Migration Monitor Guide* that you can find in the [SAP Help Portal](#).

4.4 Time analyzer (MIGTIME)

To have a baseline for optimization, determine an overview of the import and export run times. For this analysis, SAP provides a toolset that is called MIGTIME. It is delivered as a software archive, MIGTIME.SAR.

After extracting the archive, up-to-date documentation is available in the file `TimeAnalyzer.pdf`.

You can find all the necessary information about how to start the scripts and to choose the parameters in the documentation. Some examples are provided for the outcome of the analysis.

The result of every analysis is a text file and optionally an HTML file with the graphical representation and an XML file as an input file for the time-join script.

4.4.1 Export times analysis

The output of an existing export time analysis can be used during the SWPM export phase to speed up the migration process with the data provided.

The following examples are taken from an `export_time.txt` file that is gathered by the `export_time` script.

This file has two parts. The first part, which is shown in Example 4-1, relates to the package export times. The second part relates to the table export times.

Example 4-1 Example export_time.txt (part 1)

```
dataDir=/export/EXP_ECC_CLONE_R005/ABAP/DATA
export=
html=
installDir=/usr/sap/sapinst/R005/migration_logfiles_exp
top=400
xml=
```

package	time	start date	end date	size MB	MB/min
VBFA-1	6:55:16	2021-06-03 16:54	2021-06-03 23:49	1688.43	4.07
0002_ZSDRRP	5:07:40	2021-06-03 16:53	2021-06-03 22:01	8352.69	27.15
VBAP-1	4:58:21	2021-06-03 16:54	2021-06-03 21:52	2939.31	9.85
CFIN_ACCIT_APP-1	4:55:12	2021-06-03 16:54	2021-06-03 21:49	3739.24	12.67
VBFA-80	4:25:56	2021-06-03 19:42	2021-06-04 00:08	1737.28	6.53
0057_APQD	4:16:53	2021-06-03 21:29	2021-06-04 01:46	6880.56	26.78
EDIDS-1	4:13:01	2021-06-03 16:54	2021-06-03 21:07	2950.09	11.66
0017_EDI40	4:06:57	2021-06-03 19:06	2021-06-03 23:13	58818.55	238.18
CFIN_ACCCR-1	4:06:15	2021-06-03 16:54	2021-06-03 21:00	3298.64	13.40

NAST-1	4:01:08	2021-06-03	16:54	2021-06-03	20:55	3350.71	13.90
0009_VBRK	4:00:46	2021-06-03	19:01	2021-06-03	23:02	15866.39	65.90
SOFFCONT1-10	3:57:49	2021-06-03	17:15	2021-06-03	21:13	49518.26	208.22
BSIS-1	3:50:29	2021-06-03	16:53	2021-06-03	20:44	3770.94	16.36
SOFFCONT1-14	3:42:40	2021-06-03	21:10	2021-06-04	00:53	49954.99	224.35
VBFS-1	3:41:20	2021-06-03	16:54	2021-06-03	20:35	2115.52	9.56
SOFFCONT1-15	3:40:27	2021-06-03	21:13	2021-06-04	00:53	49999.72	226.81
SOFFCONT1-13	3:40:03	2021-06-03	21:08	2021-06-04	00:48	49127.76	223.26
CDCLS-1	3:39:27	2021-06-03	16:53	2021-06-03	20:33	6889.76	31.40
0001_S502	3:39:18	2021-06-03	16:53	2021-06-03	20:33	7289.13	33.24

For every package, you receive the following information:

package	Name of the package that was exported; can also be a single table, if you used package splitting
time	Runtime needed to export the package
start date	Date and time when the export of the package started
end date	Date and time when the export of the package ended
size MB	Size of the exported package in MB (for example, on disk)
MB/min	Export rate in MB per minute and is related to the export file size

The list is sorted by the export time in descending order. The list provides an overview of which packages need the most time.

Example 4-2 shows the table export times shown in the second part of the file. The top, h, or m options of the export_time script control how many tables are shown.

Example 4-2 Example export_time.txt (part 2)

table	package	time	create	data	index	merge	unload
VBFA	VBFA-1	6:55:15	0:00:00	6:55:15	0:00:00	0:00:00	0:00:00
ZSDRRP	0002_ZSDRRP	5:07:39	0:00:00	5:07:39	0:00:00	0:00:00	0:00:00
VBAP	VBAP-1	4:58:20	0:00:00	4:58:20	0:00:00	0:00:00	0:00:00
CFIN_ACCIT_APP	CFIN_ACCIT_APP-1	4:55:10	0:00:00	4:55:10	0:00:00	0:00:00	0:00:00
VBFA	VBFA-80	4:25:54	0:00:00	4:25:54	0:00:00	0:00:00	0:00:00
APQD	0057_APQD	4:16:51	0:00:00	4:16:51	0:00:00	0:00:00	0:00:00
EDIDS	EDIDS-1	4:12:59	0:00:00	4:12:59	0:00:00	0:00:00	0:00:00
EDI40	0017_EDI40	4:06:55	0:00:00	4:06:55	0:00:00	0:00:00	0:00:00
CFIN_ACCCR	CFIN_ACCCR-1	4:06:14	0:00:00	4:06:14	0:00:00	0:00:00	0:00:00
NAST	NAST-1	4:01:07	0:00:00	4:01:07	0:00:00	0:00:00	0:00:00
VBRK	0009_VBRK	4:00:45	0:00:00	4:00:45	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-10	3:57:48	0:00:00	3:57:48	0:00:00	0:00:00	0:00:00
BSIS	BSIS-1	3:50:28	0:00:00	3:50:28	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-14	3:42:39	0:00:00	3:42:39	0:00:00	0:00:00	0:00:00
VBFS	VBFS-1	3:41:19	0:00:00	3:41:19	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-15	3:40:26	0:00:00	3:40:26	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-13	3:40:02	0:00:00	3:40:02	0:00:00	0:00:00	0:00:00
CDCLS	CDCLS-1	3:39:25	0:00:00	3:39:25	0:00:00	0:00:00	0:00:00
S502	0001_S502	3:39:16	0:00:00	3:39:16	0:00:00	0:00:00	0:00:00
VBPA	VBPA-1	3:38:15	0:00:00	3:38:15	0:00:00	0:00:00	0:00:00
LIKP	0011_LIKP	3:36:18	0:00:00	3:36:18	0:00:00	0:00:00	0:00:00
BSAS	0010_BSAS	3:34:41	0:00:00	3:34:41	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-12	3:31:27	0:00:00	3:31:27	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-11	3:30:35	0:00:00	3:30:35	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-16	3:29:37	0:00:00	3:29:37	0:00:00	0:00:00	0:00:00
SOFFCONT1	SOFFCONT1-3	3:26:03	0:00:00	3:26:03	0:00:00	0:00:00	0:00:00

For every table listed you receive the following information:

- table** Name of the table exported
- package** Name of the package the table is in (when split out, table name and package name are identical)
- time** Runtime needed to export the table
- other columns** Related only to and shown during the import, such as index creation

By using this information, you can easily identify the tables that contribute the most to the export time of a package consisting of more than one table. You then might consider further export tuning activities for those tables.

You get the graphical representation of the first part of the `export_time.txt` when you use the HTML option of the `export_time` script to generate a Package Time Diagram. Figure 4-3 shows a part of such a diagram.

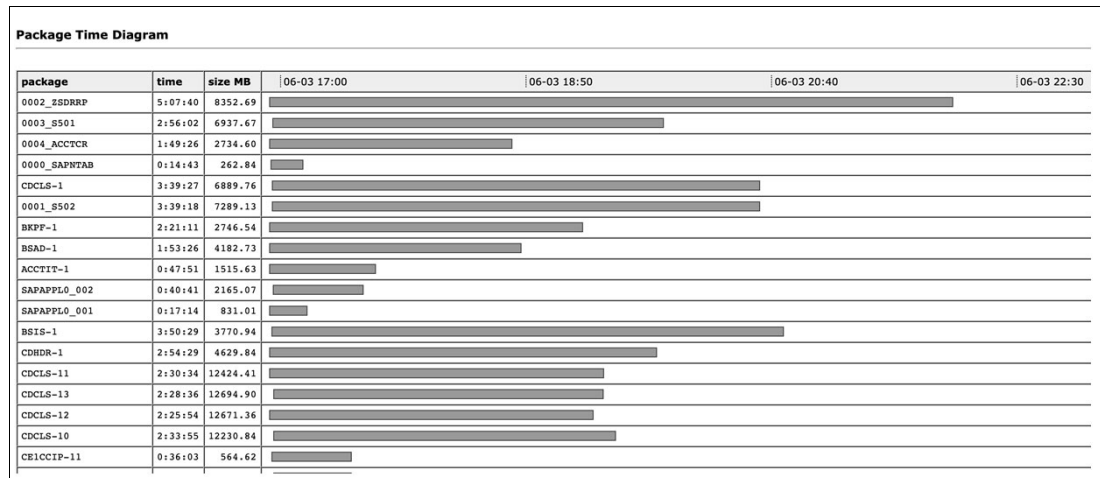


Figure 4-3 Example of an export package time diagram

The diagram is sorted by the start time of the exported package and therefore provides a chronological overview of the export. The length of the bars represents the export time.

4.4.2 Import times analysis

The following examples are taken from an `import_time.txt` file gathered by the `import_time` script.

This file has two parts. The first part relates to the package import times. The second part relates to the table import times. Part one is shown in Example 4-3.

Example 4-3 Example import_time.txt (part 1)

```
dataDir=/export/EXP_ECC_CLONE_R005/ABAP/DATA
h=1
html=
import=
installDir=/usr/sap/sapinst/R005/migration_logfiles_imp
top=
xml=
```

package	time	start date	end date	size MB	MB/min
VBOX__DT	17:49:36	2021-06-04 14:04	2021-06-05 07:53		
VBFA__DT	8:14:31	2021-06-04 14:29	2021-06-04 22:44		
VBRP__DT	5:08:06	2021-06-04 13:36	2021-06-04 18:44		
FAGLFLEXA__DT	3:45:54	2021-06-04 02:38	2021-06-04 06:23		
COEP__DT	3:18:29	2021-06-04 11:50	2021-06-04 15:09		
0014_JEST	3:10:52	2021-06-03 21:15	2021-06-04 00:25	6564.81	34.39
0009_VBRK	2:22:07	2021-06-03 22:09	2021-06-04 00:31	15866.39	111.64
VBFS__DT	2:19:10	2021-06-04 04:25	2021-06-04 06:44		
0002_ZSDRRP	2:14:27	2021-06-03 21:02	2021-06-03 23:17	8352.69	62.12
VBPA__DT	2:04:08	2021-06-04 03:18	2021-06-04 05:22		
0011_LIKP	1:58:41	2021-06-03 22:16	2021-06-04 00:14	10719.99	90.32
0010_BSAS	1:49:29	2021-06-03 21:54	2021-06-03 23:44	10980.55	100.29
CDHDR__DT	1:43:31	2021-06-03 22:27	2021-06-04 00:10		
GLPCA__DT	1:43:24	2021-06-04 03:16	2021-06-04 04:59		
0013_VRPMA	1:39:24	2021-06-03 20:41	2021-06-03 22:20	5672.43	57.07
NAST__DT	1:39:19	2021-06-04 02:59	2021-06-04 04:38		
CE1CCIP__DT	1:38:05	2021-06-04 07:54	2021-06-04 09:32		
VAPMA__DT	1:37:12	2021-06-03 21:56	2021-06-03 23:34		
0020_VLPMA	1:35:45	2021-06-03 20:33	2021-06-03 22:09	7322.53	76.48
0018_CHVW	1:35:05	2021-06-03 21:17	2021-06-03 22:52	7828.94	82.34
0001_S502	1:35:02	2021-06-03 19:33	2021-06-03 21:08	7289.13	76.70
BSIS__DT	1:34:51	2021-06-04 04:00	2021-06-04 05:35		
0007_S033	1:31:44	2021-06-03 20:22	2021-06-03 21:54	7432.97	81.03
0006_LTAP	1:30:49	2021-06-03 21:15	2021-06-03 22:45	12185.81	134.18
LIPS__DT	1:29:42	2021-06-04 08:34	2021-06-04 10:04		
0003_S501	1:29:08	2021-06-03 18:50	2021-06-03 20:19	6937.67	77.83
0028_CMFP	1:28:32	2021-06-03 22:45	2021-06-04 00:14	5774.55	65.22
0025_VEKP	1:27:05	2021-06-03 22:45	2021-06-04 00:13	8404.73	96.51
VBUP__DT	1:23:29	2021-06-04 02:14	2021-06-04 03:38		

For every package, you receive the following information:

- package** Name of the package that was imported (may also be only one table, you used package splitting)
- time** Runtime needed to import the package (including index creation)
- start date** Date and time when the import of the package started
- end date** Date and time when the import of the package ended
- size MB** Size of the imported package in MB on disk
- MB/min** Import rate in MB per minute and is related to the export file size

The list is sorted in descending order by the import time. The list provides an overview of which packages need the most time.

Example 4-4 shows the table that is related to the second section of the file. The top, h, or m options of the import_time script control how many tables are shown.

Example 4-4 Example of import_time.txt (part 2)

table	package	time	create	data	index	merge	unload
VBOX	VBOX__DT	17:49:35	0:00:00	0:00:00	17:49:35	0:00:00	0:00:00
VBFA	VBFA__DT	8:14:30	0:00:00	0:00:00	8:14:30	0:00:00	0:00:00
VBRP	VBRP__DT	5:08:05	0:00:00	0:00:00	5:08:05	0:00:00	0:00:00
FAGLFLEXA	FAGLFLEXA__DT	3:45:53	0:00:00	0:00:00	3:45:53	0:00:00	0:00:00

COEP	COEP_DT	3:18:28	0:00:00	0:00:00	3:18:28	0:00:00	0:00:00
JEST	0014_JEST	3:04:14	0:00:01	1:56:45	1:07:28	0:00:00	0:00:00
VBFS	VBFS_DT	2:19:09	0:00:00	0:00:00	2:19:09	0:00:00	0:00:00
VBRK	0009_VBRK	2:08:35	0:00:01	1:28:59	0:39:35	0:00:00	0:00:00
VBPA	VBPA_DT	2:04:07	0:00:00	0:00:00	2:04:07	0:00:00	0:00:00
ZSDRRP	0002_ZSDRRP	2:03:28	0:00:01	1:32:01	0:31:26	0:00:00	0:00:00
LIKP	0011_LIKP	1:48:02	0:00:00	1:20:40	0:27:22	0:00:00	0:00:00
CDHDR	CDHDR_DT	1:43:29	0:00:00	0:00:00	1:43:29	0:00:00	0:00:00
GLPCA	GLPCA_DT	1:43:23	0:00:00	0:00:00	1:43:23	0:00:00	0:00:00
NAST	NAST_DT	1:39:18	0:00:00	0:00:00	1:39:18	0:00:00	0:00:00
BSAS	0010_BSAS	1:38:13	0:00:00	1:19:16	0:18:57	0:00:00	0:00:00
CE1CCIP	CE1CCIP_DT	1:38:04	0:00:00	0:00:00	1:38:04	0:00:00	0:00:00
VAPMA	VAPMA_DT	1:37:11	0:00:00	0:00:00	1:37:11	0:00:00	0:00:00
BSIS	BSIS_DT	1:34:50	0:00:00	0:00:00	1:34:50	0:00:00	0:00:00
VRPMA	0013_VRPMA	1:33:01	0:00:01	1:03:46	0:29:14	0:00:00	0:00:00
VLPMA	0020_VLPMA	1:30:00	0:00:00	0:53:46	0:36:14	0:00:00	0:00:00
LIPS	LIPS_DT	1:29:41	0:00:00	0:00:00	1:29:41	0:00:00	0:00:00
CHVW	0018_CHVW	1:28:01	0:00:01	0:56:35	0:31:25	0:00:00	0:00:00
S502	0001_S502	1:27:21	0:00:00	0:58:11	0:29:10	0:00:00	0:00:00
S033	0007_S033	1:25:01	0:00:00	0:54:28	0:30:33	0:00:00	0:00:00
VBUP	VBUP_DT	1:23:28	0:00:00	0:00:00	1:23:28	0:00:00	0:00:00
S501	0003_S501	1:21:41	0:00:00	0:54:16	0:27:25	0:00:00	0:00:00
LTAP	0006_LTAP	1:21:38	0:00:01	1:10:45	0:10:52	0:00:00	0:00:00
ACCTCR	0004_ACCTCR	1:19:41	0:00:00	1:04:12	0:15:29	0:00:00	0:00:00
ZCRM_AL_I	ZCRM_AL_I_DT	1:18:19	0:00:00	0:00:00	1:18:19	0:00:00	0:00:00
VEKP	0025_VEKP	1:17:51	0:00:00	0:54:51	0:23:00	0:00:00	0:00:00
CMFP	0028_CMFP	1:17:17	0:00:00	0:55:25	0:21:52	0:00:00	0:00:00
VFSI	0008_VFSI	1:16:38	0:00:01	1:03:12	0:13:25	0:00:00	0:00:00
CDCLS	CDCLS_DT	1:13:30	0:00:00	0:00:00	1:13:30	0:00:00	0:00:00
VBAP	VBAP_DT	1:12:27	0:00:00	0:00:00	1:12:27	0:00:00	0:00:00
QAMV	0015_QAMV	1:12:02	0:00:01	1:06:45	0:05:16	0:00:00	0:00:00
JSTO	0033_JSTO	1:11:00	0:00:00	0:56:32	0:14:28	0:00:00	0:00:00
ONRVB	0030_ONRVB	1:10:44	0:00:00	0:57:23	0:13:21	0:00:00	0:00:00
ZSDT_LOG	0024_ZSDT_LOG	1:06:03	0:00:00	0:54:01	0:12:02	0:00:00	0:00:00
CE4CCIP	CE4CCIP_DT	1:04:16	0:00:00	0:00:00	1:04:16	0:00:00	0:00:00
CFIN_ACCIT	CFIN_ACCIT_DT	1:03:25	0:00:00	0:00:00	1:03:25	0:00:00	0:00:00
ZSDRRK	0016_ZSDRRK	1:00:51	0:00:01	0:54:09	0:06:41	0:00:00	0:00:00
CKMI1	0021_CKMI1	1:00:13	0:00:00	0:44:24	0:15:49	0:00:00	0:00:00
		94:02:43	0:00:09	24:00:22	70:02:12	0:00:00	0:00:00

For every table listed you receive the following information:

table	Name of the table exported
package	Name of the package the table is in; when split out, table name and package name are identical
time	Total runtime needed to import and create the index
data	Runtime needed to import the table data only
index	Runtime needed to create all indexes
other columns	Merge and unload are only HANA related

By using this information, you can identify the tables that contribute the most to the import time of a package that comprises more than one table.

You get the graphical representation of the first part of the `import_time.txt` when you use the HTML option of the `import_time` script to generate a package time diagram. Figure 4-4 shows a part of such a diagram.



Figure 4-4 Example of an import package time diagram

The diagram is sorted by the start time of the imported package and therefore gives a chronological overview of the import. The length of the bars represents the import time. The dashed lines for table D010TAB mean that the import was interrupted and continued later.

4.4.3 Time join analysis

The third option of the Time Analysis Tool is to join the export and import times. This produces the file `time_join.txt`. An example of this file is shown in Figure 4-5. Choosing the HTML option also produces the file `time_join.html` with the graphical representation of the text version.

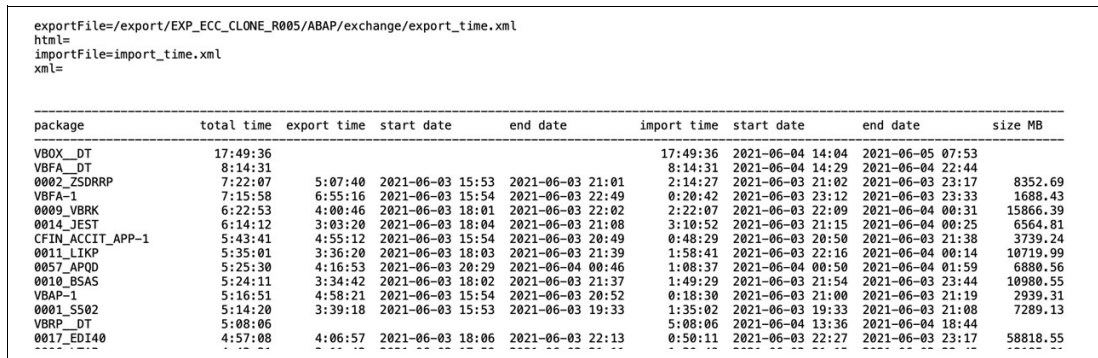


Figure 4-5 Example of `time_join.txt`

Note that in Figure 4-5 the records are in one line inside the text file.

Figure 4-6 on page 53 shows the graphical representation of the Package Join Time Diagram.

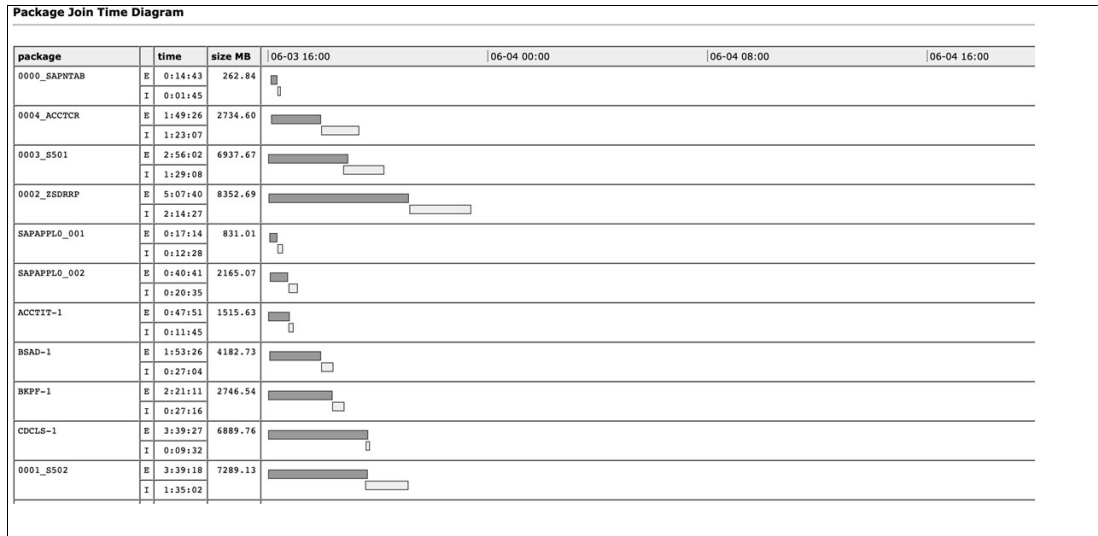


Figure 4-6 Example of a package join time diagram

4.5 Tools for package and table splitting

To improve the speed of the SAP system copy, multiple R3load processes can export and import data in parallel. Different options are available to package the data during export and import as shown in Figure 4-7.

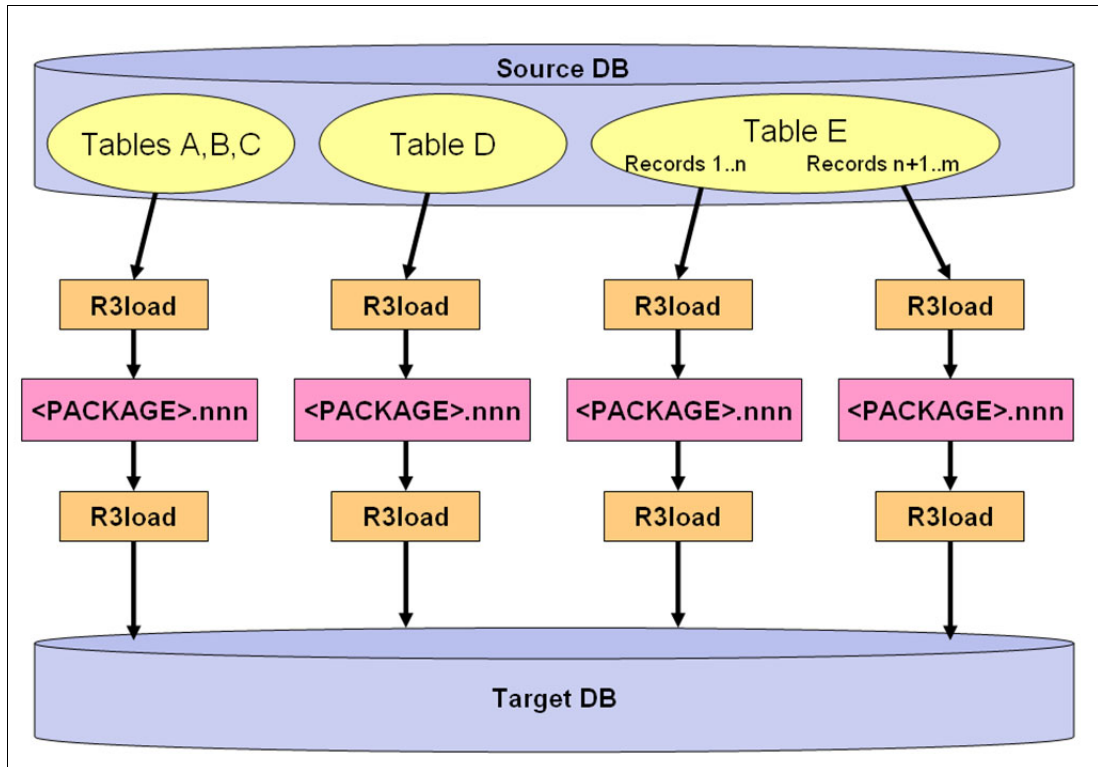


Figure 4-7 Package and Table Splitting Options

As shown in Figure 4-7, the contents of a package can vary:

- ▶ A package can contain data from multiple tables as demonstrated by tables A, B, and C.
- ▶ A package can contain all of the data from a single table as shown by table D.
- ▶ A package can contain only a subset of a table as is shown by table E.

To export a single table into multiple data packages, the R3load process requires a WHERE file which defines the range of records that should be extracted from the table.

In most cases, the source database contains a small set of large tables. You can export these tables in parallel to dedicated package files where each file contains the data of one table. For large tables, performance can be improved by exporting and importing a single table with multiple R3load processes in parallel.

SAP provides several tools for splitting packages or tables:

- ▶ Package splitter

Package Splitter splits tables from existing structure files. For example, you can explicitly specify the tables which should be split into separate structure files. You can also split tables into separate packages if their size exceeds a configurable threshold. Additional split options are available.
- ▶ R3ta/SAPuptool

This tool can generate multiple WHERE conditions for a table which can then be used to export the data of this table with multiple R3load processes in parallel. Each R3load process requires a WHERE condition to select only a subset of the data in the table.
- ▶ Software Provisioning Manager (SWPM) as part of the SL Toolset

This tool is used for the table splitting preparation. It invokes R3ta/SAPuptool and the Package Splitter and automates some of the tasks which must otherwise be performed manually.

4.6 SAP Migration Checker

The Migration Checker (migcheck) pack includes 3 different tools:

- ▶ Package Checker
- ▶ Object Checker
- ▶ Table Checker

Based on the selected run option of migcheck, the checker can be run on the target database host to verify the import has been performed successfully using different analysis strategies and deep investigation.

The Package Checker is used to validate that all packages are successfully loaded. It validates only the TSK files and does not compare the actual tables within the database itself.

The Object Checker is used to verify that all objects, which include tables, views, indexes, and primary keys, are successfully created and loaded in the database, based on log files and TSK files only.

The Table Checker will validate the TOC file row values against the actual database table records and shows differences. A sample output is shown in Example 4-5 on page 55.

For more information, see the following SAP Notes:

- 1900454 - Check amount of table rows after a System Copy [How To]
- 2009651 - Table Checker Tool as Part of a System Copy with R3load

Example 4-5 Example for Table Checker

```
cd /tmp/migration_checker
setenv JDBC_JAR /db2/db2tar/sqllib/java/db2jcc4.jar
setenv JAVA_HOME /db2/db2tar/sqllib/java/jdk64/jre

example: table_checker_cmd.properties file

tocDirs=/tmp/ibmas1/tmpexp1/ABAP/DATA
driver=com.ibm.db2.jcc.DB2Driver
unicode
numberOfDBConnections=100
url=jdbc:db2://localhost:5912/TAR:currentSchema=SAPTAR;deferPrepares=0;connectionCloseWithi
nFlightTransaction=2;
user=db2tar
password=mygoodpassword./table_checker.sh

./table_checker.sh
+ /db2/db2tar/sqllib/java/jdk64/jre/bin/java -cp
./../migcheck.jar:/db2/db2tar/sqllib/java/db2jcc4.jar com.sap.inst.lib.app.SecureStartup
' com.sap.inst.migcheck.TableChecker
Table 'DDNTF_HIST' has different row count: 20330 and 20365.
Table 'DDNTF' has different row count: 1316922 and 1327594.

db6x141001:db2tar 55> cat table_checker.log
INFO: 2022-03-03 11:19:17
Table Checker is started.

CONFIG: 2022-03-03 11:19:17
Application options:
driver=com.ibm.db2.jcc.DB2Driver
numberOfDBConnections=100
password=*****
tocDirs=/tmp/ibmas1/tmpexp1/ABAP/DATA
unicode=
url=jdbc:db2://localhost:5912/TAR:currentSchema=SAPTAR;deferPrepares=0;connectionCloseWithi
nFlightTransaction=2;
user=db2tar

CONFIG: 2022-03-03 11:19:17
Table mapping for TOC file tables:
DDNTF_1B -> DDNTF_CONV_UC
DDXTF_1B -> DDXTF_CONV_UC
DDNTF_2B -> DDNTF
DDNTT_1B -> DDNTT_CONV_UC
DDXTF_2B -> DDXTF
DDXTT_1B -> DDXTT_CONV_UC
DDNTT_2B -> DDNTT
DDXTT_2B -> DDXTT

INFO: 2022-03-03 11:19:17
Table 'DDNTF' is used instead of TOC file table 'DDNTF_2B'.

INFO: 2022-03-03 11:19:17
Table 'DDNTF_CONV_UC' is used instead of TOC file table 'DDNTF_1B'.

INFO: 2022-03-03 11:19:17
Table 'DDNTT' is used instead of TOC file table 'DDNTT_2B'.

INFO: 2022-03-03 11:19:17
```

Table 'DDNTT_CONV_UC' is used instead of TOC file table 'DDNTT_1B'.

INFO: 2022-03-03 11:19:17

Table 'DDXTF' is used instead of TOC file table 'DDXTF_2B'.

INFO: 2022-03-03 11:19:17

Table 'DDXTF_CONV_UC' is used instead of TOC file table 'DDXTF_1B'.

INFO: 2022-03-03 11:19:17

Table 'DDXTT' is used instead of TOC file table 'DDXTT_2B'.

INFO: 2022-03-03 11:19:17

Table 'DDXTT_CONV_UC' is used instead of TOC file table 'DDXTT_1B'.

WARNING: 2022-03-03 11:19:25

Table 'DDNTF_HIST' has different row count: 20330 and 20365.

WARNING: 2022-03-03 11:19:25

Table 'DDNTF' has different row count: 1316922 and 1327594.

ERROR: 2022-03-03 11:19:49

134491 tables are processed.

2 tables have different row count on source and target systems.

Tables with different row count are saved in 'invalid_tables.txt' file.

INFO: 2022-03-03 11:19:49

Table Checker is stopped.

4.7 SAP data definition tools

R3load creates the database tables during the import. The tool supports the common SQL Data Definition Language (DDL) elements for different database management engines. For special table types, R3load can use .SQL files to support these tables. SAP has developed two tools to support checking whether special tables exist on the source system.

SMIGR_CHECK_DB6 does some checks and provides a result report to be considered on the target. SMIGR_CREATE_DDL creates Db2 specific .SQL files that are required for the target. The SMIGR_CHECK_DB6 report also checks for inconsistencies in the mapping of tables to tablespaces.

Both reports can be run using SAP transaction SE38. Run SMIGR_CHECK_DB6 first, followed by SMIGR_CREATE_DDL if you choose this option.

4.7.1 SMIGR_CHECK_DB6

During the lifetime of the system, database administrators might introduce Db2 database-specific features to optimize the data storage as in the following examples:

- ▶ Modified inline length for large objects (LOB)
- ▶ Assignment of tables to dedicated tablespaces

- ▶ Usage of special table types like:
 - Range partitioned tables
 - Multidimensional clustering tables
 - Insert time clustering tables
 - Column-organized tables

Sometimes, there is a mismatch between the data placement in the database and the parameters that are defined within the SAP Data Dictionary. One example is the placement of a table without changing the SAP Data Class assignment. In such a case, the table is created according to the SAP assignment, and as a result, the table will not be created in a dedicated tablespace during migration. This is true when range partitioned tables exist and their data definition does not match the SAP boundary conditions.

Therefore, the SAP report SMIGR_CHECK_DB6 looks for these special cases and generates a summary that can be used to fix existing issues before the start of the migration. To allow for enough time to apply changes and fixes without time pressure, run this check early in the process before the migration.

The report provides the following information:

- ▶ Existence of range partitioned tables and validity of setup
- ▶ Mismatch between data class assignment and physical table location
- ▶ Tables with modified LOB inline length
- ▶ Number of insert time clustering (ITC) Tables, multi dimensional clustering (MDC) tables, and column-organized (CDE) tables.
- ▶ Status of concurrency threshold of workload management (WLM)

The result of the report reflects special Db2 objects and checks the potential data placement on the target system.

Impact of WLM concurrency threshold setting

The WLM concurrency threshold setting can also affect the export of the system. When the concurrency threshold is active, the R3load processes can be queued at a database level, which might affect parallel R3load export processes, which might increase the export times.

The WLM concurrency threshold is active when tables of type *column-organized* exist in the source system. This can include objects that have been created for test purposes but are not actively used anymore.

Data class mismatch

A potential data class mismatch due to invalid configuration might lead to data corruption or data loss.

To protect yourself from these issues, run the report before each heterogeneous system copy project and resolve problems that are reported before you start a migration.

The report can export the results to a file that can be used as a record about actions that are required and what actions were taken as preparation. For more details about the report, including the prerequisites, see SAP Note 3246738 - DB6: Migration Check Tool SMIGR_CHECK_DB6.

4.7.2 SMIGR_CREATE_DDL

SMIGR_CREATE_DDL creates DDL files to support the creation of Db2-specific objects in the target database. Run this report on the source system before exporting the database.

The report was designed for SAP NetWeaver Business Warehouse related special tables, but it was extended to address ERP specific enhancements, too. These enhancements are support for range partitioned tables and user customized LOB inline sizes

Both table types can exist on the source system, and if no .SQL files exist, the specific settings for those tables will not be migrated, and the tables are created with the SAP default settings

For example, if the system uses range partitioned tables to overcome the 4 TB LOB size limit, you might see the error ARCHITECTURAL PAGE LIMIT REACHED because partitioning information is missing in the target system and because the tables are created without partitions.

For user customized LOB inline sizes, the system does not generate an immediate error, but performance might degrade if you introduced a different LOB inline size for certain tables.

For the report SMIGR_CREATE_DDL, you must specify a location for the generated .SQL files. During SWPM export, you are prompted for this location, too. During the SWPM export processing, the .SQL files are copied to the correct location for the import and automatically used during import. Figure 4-8 shows the inputs that are generated for R3load to customize the target database.

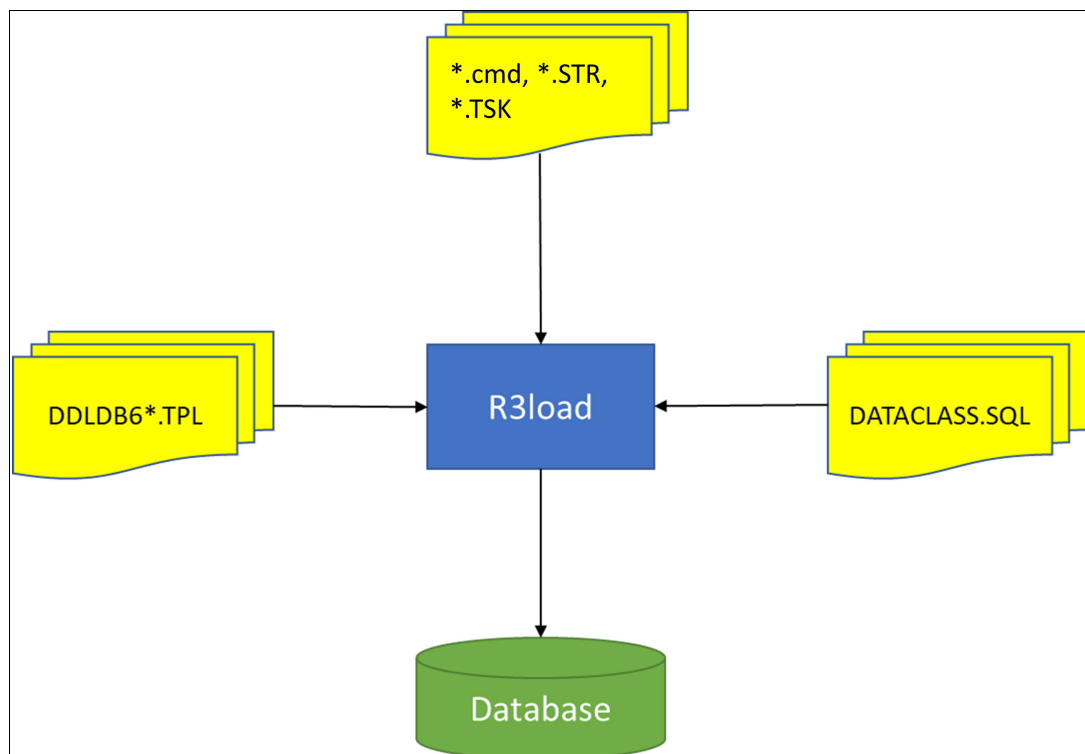


Figure 4-8 Input Files to R3load for Importing the Target Database

If you do not want certain objects to be created with the source database setting, you must delete the tables manually from the .SQL files.

The format for the generated DDL files is shown in Example 4-6.

Example 4-6 DDL File Example

```
tab: <Table name>
sql: <DDL Statement(s) for table>
ind: <Index name>
sql: <DDL Statement(s) for index>
```

The files are created with the following naming convention:

<dataclass>.SQL

R3load can use these DDL files to create database objects that are based on the syntax provided in the file.

Example 4-7 shows the input files that are used by R3load. The example was generated for a standard E fact table of an InfoCube.

Example 4-7 DDL File for an E Fact Table

```
tab: /BIC/EBFABCUBE5
sql: CREATE TABLE "/BIC/EBFABCUBE5"
("KEY_BFABCUBE5P" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE5T" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE5U" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE51" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE52" INTEGER
DEFAULT 0 NOT NULL,
"/BIC/BCRMEM_QT" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL,
"/BIC/BCRMEM_VA" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BINVCD_VA" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BINVCD_QT" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL,
"/BIC/BRTNSVAL" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BRTNSQTY" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL)
IN "&location&"
INDEX IN "&locationI&"
LONG IN "&locationL&"
PARTITIONING KEY ( "KEY_BFABCUBE51"
, "KEY_BFABCUBE52"
, "KEY_BFABCUBE5T"
, "KEY_BFABCUBE5U"
)
USING
HASHING;
ALTER TABLE "/BIC/EBFABCUBE5" LOCKSIZE ROW;
ind: /BIC/EBFABCUBE5~0
```

```
ind: /BIC/EBFABCUBE5~01
ind: /BIC/EBFABCUBE5~03
ind: /BIC/EBFABCUBE5~P
sql: CREATE UNIQUE INDEX "/BIC/EBFABCUBE5~P" on "/BIC/EBFABCUBE5"
("KEY_BFABCUBE5T",
"KEY_BFABCUBE51",
"KEY_BFABCUBE52",
"KEY_BFABCUBE5U",
"KEY_BFABCUBE5P")
CLUSTER
ALLOW REVERSE SCANS;
```

Note: The report SMIGR_CREATE_DDL can run for several hours as a dialog process. To successfully run the report, ensure the SAP profile parameter `rdisp/max_wprun_time` is set to at least 7200 or 0 for no timeout. You can set it temporarily using SAP transaction RZ11

For more details about the report including the prerequisites, refer to *SAP Note 3208238 - DB6: Enhancements to SMIGR_CREATE_DDL for range partitioning and inline LOB size*.



Export optimization techniques

This chapter describes different techniques to ensure a smooth process flow and improve export performance during a heterogeneous system copy. The optimizations of unsorted export and package splitting are default optimizations. Others such as Db2 query block prefetch, export server setup, and the R3load parameter **-compress** are new or intended to be used in special situations only.

This chapter includes the following topics:

- ▶ 5.1, “Activities to be done before export” on page 66
- ▶ 5.2, “Unsorted versus sorted export” on page 66
- ▶ 5.3, “R3load export dump compression” on page 71
- ▶ 5.4, “Package splitting” on page 74
- ▶ 5.5, “Local or remote export server scenarios” on page 76
- ▶ 5.6, “Db2 query block prefetch feature” on page 79
- ▶ 5.7, “Db2 workload management” on page 80
- ▶ 5.8, “Other Db2 features” on page 81

5.1 Activities to be done before export

Before you migrate from any database to Db2 LUW, a few preparation steps must be performed. Planning and performing those steps safeguards the entire installation and migration approach. With Db2 and SAP, you can run older SAP releases and use newer versions of Db2. It is worth considering running a recent Db2 version in your target system to take advantage of new capabilities in the newer Db2 releases. Before starting any migration, consult the SAP product availability matrix to verify that you are running compatible versions. The SAP product availability matrix (PAM) shows which SAP kernel, SAP release, and database versions are supported.

It is also important to validate the current installation prerequisites for your Db2 software installation, validate operating system kernel settings, and look for known issues. It is important to use only a certified SAP/Db2 software image for the installation. All other Db2 images, including images from IBM, are not supported.

You must have the skills available and a thorough understanding of the source and target database to make a system copy. To run the migration, a certified OS/DB migration consultant is mandatory. For more information, see *SAP Note 82478 - SAP system OS/DB migration*. To ensure a successful production migration, it is mandatory to use an SAP OS/DB migration check service.

Consult the following SAP Notes and Guides before performing any technical activity:

- ▶ *System Copy for SAP Systems Based on the Application Server Java of SAP NetWeaver 7.5, and SAP Solution Manager 7.2 SR2 Java, on UNIX*
- ▶ *Database Migration Option: Target Database IBM Db2 for Linux, UNIX, and Windows*
- ▶ *Database Administration Guide for SAP on IBM Db2 for Linux, UNIX, and Windows*
- ▶ *SAP Note 82478 - SAP system OS/DB migration.*
- ▶ *SAP Note 101809 - DB6: Supported Db2 Versions and Fix Pack Levels*
- ▶ *SAP Note 1718576 - Migration from SAP HANA to another database system*
- ▶ *SAP Note 888210 - NW 7.**: System copy (supplementary note)*
- ▶ *SAP Note 816773 - DB6: Installing the Application-Specific Db2 License from SAP*
- ▶ *SAP Note 1680045 - Release Note for Software Provisioning Manager 1.0*

Guides can be found in the [Guide Finder for SAP NetWeaver section](#) of the SAP Help Portal.

Note: Before you copy the SAP system to Db2, implement the following SAP code corrections in the source SAP system:

- ▶ *SAP Note 2267446 - DB6: Support of tablespace pools.*
- ▶ *SAP Note 1456402 - DB6: DBA Cockpit Correction Collection SAP Basis 7.02 / 7.30 / 7.31 / 7.40.*
- ▶ *SAP Note 3246738 - DB6: Migration Check Tool SMIGR_CHECK_DB6*
- ▶ *SAP Note 3208238 - DB6: Enhancements to SMIGR_CREATE_DDL for range partitioning and inline LOB size.*

5.2 Unsorted versus sorted export

With a sorted export, the rows of a table are read in the sequence of the primary key. If the cluster ratio is not optimal, data pages are read continuously from disk. In addition, database sort operations might occur, which adds to the export time. These issues prevent you from

achieving an optimal runtime. For more information, see *SAP Note 954268 - Optimization of export: Unsorted unloading*.

5.2.1 Runtime comparison

With an unsorted export, the pages are read continuously and the export generally takes less time than a sorted export. This improvement can be significant, so an unsorted export can be recommended for most of your tables. However, when an index allows index-only access, a sorted export might be faster than an unsorted export, especially if the data is fragmented.

Table clusters have some boundary conditions that prevent unsorted exports. The unsorted export is also not possible if a code page conversion takes place during the migration or if declustering is performed. Table clusters are discussed in 5.2.5, “Table clusters and cluster tables” on page 69. For other exceptions regarding sorted export, see *SAP Note 954268*.

Note: The unsorted export is a powerful optimization to improve the export time. However, the data is also imported unsorted, so the database performance might not be optimized in the target system. You might want to plan for a subsequent reorganization of unsorted tables in the target system.

Figure 5-1 shows improved runtime in minutes with unsorted export. The improvement is typically 30%–50% but can be more. For table SOC3, it is only about 10%, so it might be faster to export the table that is sorted to optimize the data placement on the target side during import.

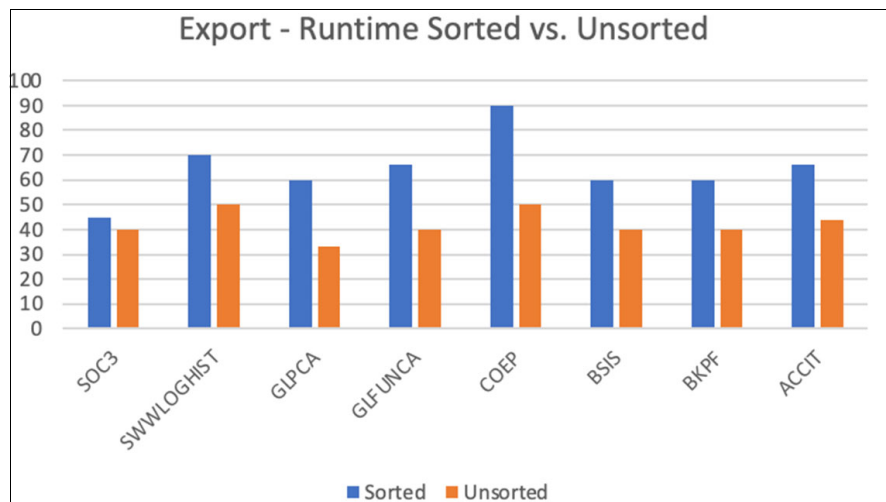


Figure 5-1 Export Runtime Comparison Results

5.2.2 Enabling unsorted export

The mode of export is controlled by the `ORDER_BY_PKEY` keyword in the `prikey`-section of the `DDL<DBS>.TPL` file. In the current example, the file is named `DDLDB6.TPL` as shown in Example 5-1.

Example 5-1 Keyword that specifies sorted export

```
prikey: AFTER_LOAD ORDER_BY_PKEY
```

If the keyword `ORDER_BY_PKEY` is deleted, the unloading is done unsorted.

When using the MigMon, you can use the `ddlMap`-option in the `export-properties` file, which names a file that contains the mapping between the package names and the two DDL template files: `DDL<DBS>.TPL` and `DDL<DBS>_LRG.TPL`.

The `DDL<DBS>_LRG.TPL` template file does not contain the `ORDER_BY_PKEY` keyword in the `prikey`-section. `R3ldctl` generates the `DDL<DBS>_LRG.TPL` template file and the `DDL<DBS>.TPL` file. You can use the MigMon DDL Template Mapping File to set up exports for different tables with or without sorting.

In Example 5-2, the packages `SAPCLUST` and `SAPSDIC` are using the `DDL<DBS>.TPL` template file, and the table `MSEG`, which has a package of its own, and the package `SAPAPPL1` are unloaded by using the `DDL<DBS>_LRG.TPL` template file.

Example 5-2 Example of a MigMon DDL Template Mapping File

```
[ SORTED UNLOAD ]
# DDL file for sorted unload
ddlFile = <path>/DDL<DBS>.TPL
#package names
SAPCLUST
SAPSDIC
[ UNSORTED UNLOAD ]
# DDL file for unsorted unload
ddlFile = <path>/DDL<DBS>_LRG.TPL
# package names (may also contain only one table)
MSEG
SAPAPPL1
```

5.2.3 Resource consumption

Monitoring data typically shows that CPU consumption with an unsorted export is slightly higher but spread over a shorter period. So, this indicates an efficient usage of resources.

The sorted export does not increase the CPU usage significantly, but during a sorted export, you might see spikes in the write operations per second. See Figure 5-2.

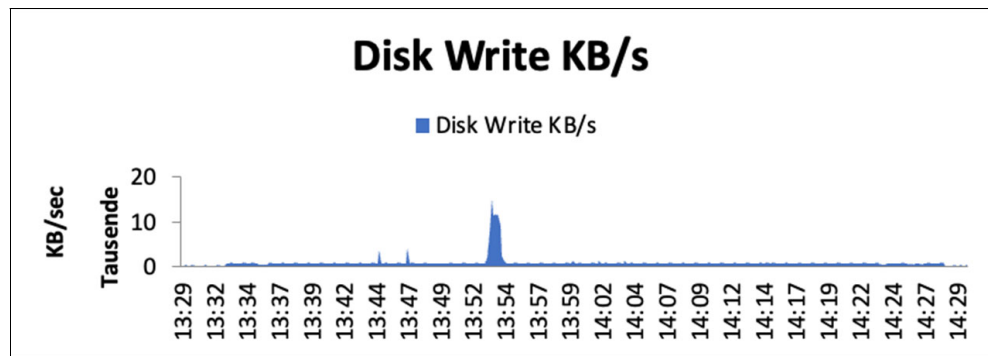


Figure 5-2 Sorted export disk writes

As shown in Figure 5-2, there is a constant write throughput that reflects the writing of the R3load dump files to disk. There is one significant peak in write operations at approximately 13:53. This is not due to an increased throughput. It is due to write operations of Db2 caused by a sorting operation that uses the disk.

Table 5-1 shows the results of database metrics during a sorted and unsorted export.

Table 5-1 Db2 Monitoring Metrics Comparison

Monitoring metric	Sorted	Unsorted
Total sorts	= 70	= 23
Total sort time (ms)	= 2528650	= 157
Sort overflows	= 4	= 0
Active sorts	= 0	= 0
Buffer pool temporary data logical reads	= 2438151	= 19
Buffer pool temporary data physical reads	= 1228390	= 0
Buffer pool data writes	= 1218386	= 0
Asynchronous pool data page writes	= 1217750	= 0
Buffer pool index logical reads	= 2095225	= 5611
Buffer pool index physical reads	= 762257	= 614
Total buffer pool read time (milliseconds)	= 21442746	= 9702406
Total buffer pool write time (milliseconds)	= 4358579	= 0
Total elapsed asynchronous write time	= 3931277	= 0
Asynchronous data read requests	= 1977833	= 3086773
Asynchronous index read requests	= 384730	= 2

The Db2 monitoring metric Total sort time (ms) shows a longer time for sorting with the sorted export. It also shows four sort overflows. The sort overflows cause the peak write workload because the sort writes to and reads from temporary tables spaces on the disk.

As more data is written and read, more buffer pool accesses are performed. Bufferpool accesses do not generate I/O but do use additional CPU cycles, which explains the increased runtime.

The monitoring data also shows significantly more index accesses when the export is sorted, which is a good indicator that sorting is involved and that an index is used for the access to the data.

5.2.4 Sorting and cluster tables

Table clusters often determine the overall migration time.

If code page conversions are performed or if the table cluster is converted to one or more transparent tables, then the table cluster must be exported as sorted. If the table cluster must be sorted and if table clusters are very large, then additional focus might be required for this table type. For example, typical candidates are CDCLS, EDI40, RFBLG.

Also, for table clusters, the process for compressing the R3load dump files and the declustering is more CPU intensive.

5.2.5 Table clusters and cluster tables

The information in this section can help you to understand why table clusters must not be unloaded as unsorted in certain situations. The discussion includes some introductory information about table clusters.

Table clusters are the physical representation of one or multiple logical tables. For example, the table cluster CDCLS contains the cluster tables CDPOS and PCDPOS. A logical row in a cluster table is mapped to one or more physical rows in the table cluster.

In this example in Figure 5-3, the definition of the table cluster DOKCLU contains just the cluster table DOKTL.

Field name	Key	DTyp	Length
ID	<input checked="" type="checkbox"/>	CHAR	2
OBJECT	<input checked="" type="checkbox"/>	CHAR	60
LANGU	<input checked="" type="checkbox"/>	CHAR	1
TYP	<input checked="" type="checkbox"/>	CHAR	1
DOKVERSION	<input checked="" type="checkbox"/>	NUMC	4
PAGENO	<input checked="" type="checkbox"/>	INT2	5
TIMESTMP	<input type="checkbox"/>	CHAR	14
PAGELG	<input type="checkbox"/>	INT2	5
VARDATA	<input type="checkbox"/>	RAW	3800

Figure 5-3 SE11 output for table-cluster DOKCLU

You can identify every logical row of a cluster table by the key fields of the table cluster. The data for the logical rows of a cluster table is stored in the column VARDATA. If the data is longer than the defined length of VARDATA, additional records for the same key are stored with an increased number of PAGENO. The field PAGELG describes the length of the data stored inside the VARDATA field. See Figure 5-4.

ID	OBJECT	LANGU	TYP	DOKVERSION	PAGENO	TIMESTMP	PAGELG
DT	RSMD_CONVERSION	D	E	0038	0	6) 3UR6zzKQLkD4	3800
DT	RSMD_CONVERSION	D	E	0038	1	6) 3UR6zzKQLkD4	3800
DT	RSMD_CONVERSION	D	E	0038	2	6) 3UR6zzKQLkD4	3800
DT	RSMD_CONVERSION	D	E	0038	3	6) 3UR6zzKQLkD4	60

Figure 5-4 Records of Table Cluster DOKCLU for one Logical Row

Figure 5-4 shows four records for the key value DT RSMD_CONVERSION D E 0038 with an increasing PAGENO. The first three records are filled with data in the field VARDATA. This can be seen from the PAGELG with a value of 3800 that matches the length of the VARDATA field.

The fourth record with PAGENO 3 has a PAGELG of 60 meaning that this record is not filled to the end and indicates the end of the rows that belong together.

During code page conversions, the contents and the length of the records might change. Even the number of the physical records belonging to a logical record might change.

Because the physical records are built together to a logical record, the data must be read in a sorted way to find all physical records that belong to a logical record. Therefore, an unsorted unload is not possible.

The same requirement for a sorted unload for files during code page conversions also includes code page conversions that are caused by a change to hardware architecture with different endianness. One example for this is, AIX on IBM Power Servers or HP-UX with PA-RISC CPU to an x86-based infrastructure like Linux. In this case, the code page changes from 4102 (big-endian) to 4103 (little-endian).

The restriction of sorted export also applies to declustering during the export. The logical records must be built first, followed by the conversion to one or more transparent table records. If no changes to the contents of the records are made, the logical record does not have to be constructed and the table cluster can be unloaded unsorted.

Based on these restrictions, it is generally not possible to use unsorted export as an optimization when using table clusters.

Note: If you specify to use an unsorted export, R3load ignores this and performs a sorted export, if necessary, and writes a message to the R3load log file.

5.3 R3load export dump compression

When dumping the exported data to files, determine the size of these files and compare the results with the size of the table in the database. It is now possible to affect the size of the export dump files to improve the export performance of the R3load. This topic includes R3load compression, which is not the same as the different options for Db2 compression during the import. R3load compression affects the data as it is written to the export files. Db2 compression options control the compression process during the import of the data into the target database.

5.3.1 Table and dump-file size comparison

Figure 5-5 shows the size of compressed export dump files of selected tables compared to the table size of those tables.

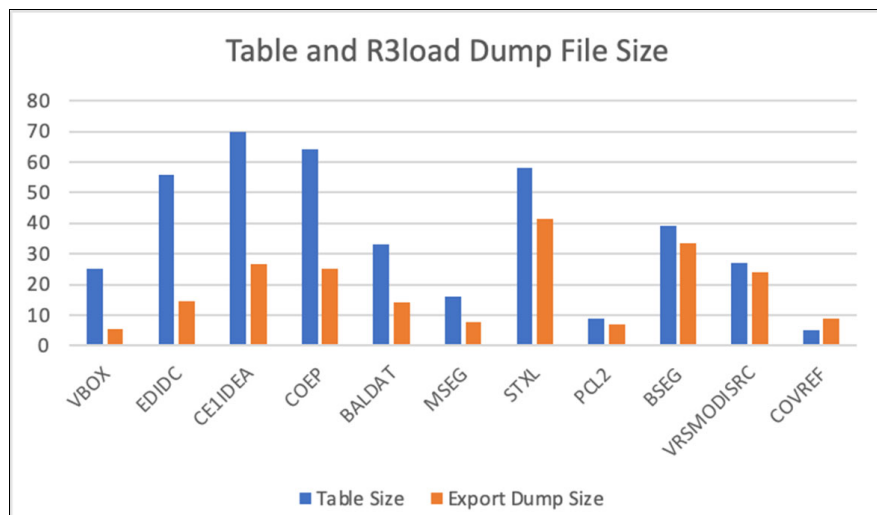


Figure 5-5 Export File Size Comparison

The size of the compressed export dump files are 20%–40% of the physical size of the table for many transparent tables. For cluster tables and INDX-like tables, the dump file size is approximately 80% of the physical table size. For some tables, the dump is even larger than the table object, which is illustrated by table COVREF in Figure 5-5.

You can use this information to assess the space required for the export dump and the amount of data that needs to be transferred from source to target.

Use the statement shown in Example 5-3 to get the allocated size of data, long field (LF), and long objects (LOB) in the database. The statement returns the size of all tables without the defined indexes.

Example 5-3 SQL to determine the Db2 Data Size

```
SELECT SUM(DATA_OBJECT_P_SIZE+LONG_OBJECT_P_SIZE+LOB_OBJECT_P_SIZE) AS TABSIZE  
FROM SYSIBMADM.ADMINTABINFO
```

Alternatively, you can combine all physical table sizes by using the SAP DBA Cockpit. You can use the panel Top Space Consumers. In the panel, combine the Phys. Data Size, Phys. Long Data Size and the Physical LOB Data size. You can also run the SQL statement that is shown in Example 5-4 on the command line to get the sizes of the largest 100 tables. The results can be used as input for splitting packages.

Example 5-4 SQL command to determine the 100 largest tables

```
SELECT TABNAME, (DATA_OBJECT_P_SIZE+LONG_OBJECT_P_SIZE+LOB_OBJECT_P_SIZE) AS  
TABSIZEINGB FROM SYSIBMADM.ADMINTABINFO ORDER BY TABSIZE DESC FETCH FIRST 100 ROWS  
ONLY
```

5.3.2 Static R3load compression

Compression of the export dump file is done by default when using R3load. Although compression can provide many benefits, it uses additional CPU resources similar to code page conversions or declustering when done by R3load.

If you look at the results, many tables benefit from the R3load compression, and so the migration process also benefits. Smaller R3load dump files mean less I/O and less data to be transferred over the network. However, for cluster tables and INDX-like tables, the compression does not seem to be effective. Often, the data cannot be compressed well because of the nature of the data. In these cases, CPU resources are used to compress the data used during the export with minimal savings in I/O or network bandwidth.

5.3.3 Adaptive R3load compression

SAP has recently introduced a new feature that uses a different compression algorithm for cluster tables and INDX-like tables.

The default algorithm for compression is based on the Lempel–Ziv–Welch (LZW) algorithm that normally delivers a lossless data compression. The new algorithm uses the Run-length encoding algorithm (RLE) which requires less compute power but typically provides less optimal compression results when compared to LZW.

5.3.4 Improvements with adaptive R3load compression

The R3load enhancement now dynamically switches to RLE-based compression for cluster or INDX-like tables. As a result, export runtime for some tables is reduced significantly.

Figure 5-6 on page 73 shows that the table CONVREF exports more than 8 times faster. The tables STXL and BALDAT also showed an improvement of a factor of 2 or more. So, this optimization might significantly reduce your export runtime.

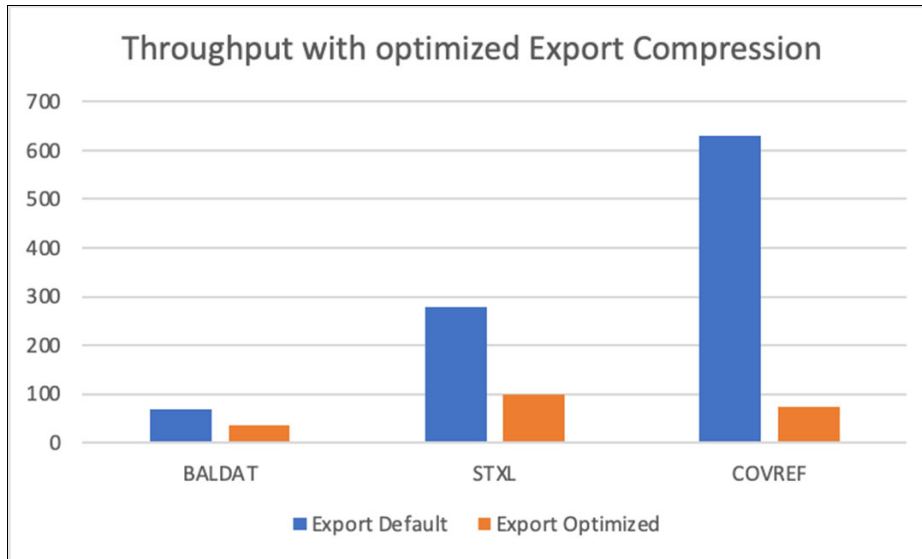


Figure 5-6 R3load Export Performance Comparison

Typically, saving on one resource type leads to increased usage in another. The first expectation is that the R3load dump file increases in size along with increased I/O rates. Figure 5-7 shows that the export dump file sizes increase 60%–80%, but often, it saves significantly on CPU resources and reduces export time.

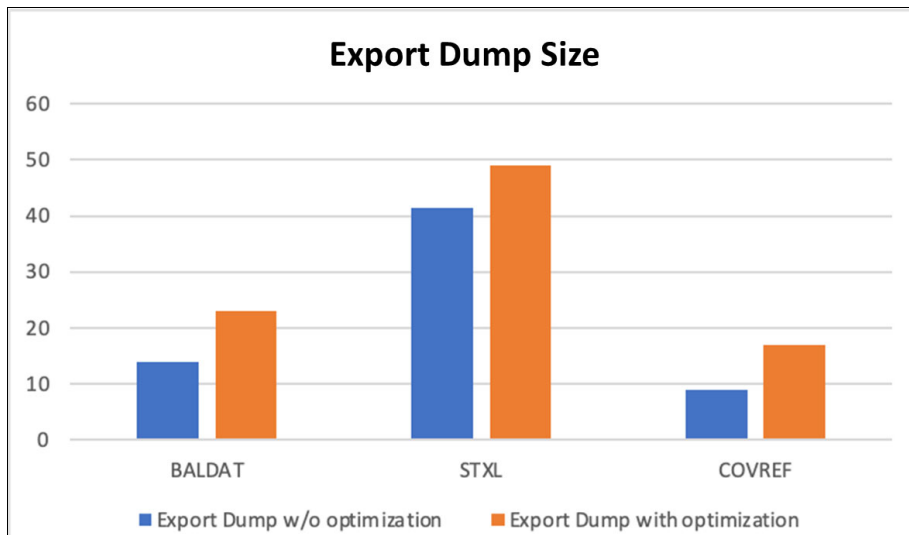


Figure 5-7 R3load Export Dump File Size Comparison

Important: File sizes can sometimes increase significantly. During one test, the export dump size for table VBOX increased from 5 GB to over 70 GB, a factor of 14.

5.3.5 Usage of adaptive R3load compression

To enable the R3load dump file optimization for compression, use the R3load parameter **-compress adapt** during the export as shown in Example 5-5.

Example 5-5 R3load example with optimized dump file compression

```
/R3load -e BALDAT.cmd -datacodepage 4103 -l BALDAT.log -continue_on_error  
-compress adapt
```

The R3load parameter **-compress** affects only the export of the data. The R3load on the import detects the dump file compression method automatically and chooses the correct option for the data decompression.

Note: The optimized compression is beneficial only for cluster and INDX-like tables and uses the appropriate algorithm automatically. Always use the **adapt** keyword for this optimization and do not use **r1e** or **1zw**. If you choose the wrong compression algorithm, the export dump size can increase significantly.

5.4 Package splitting

All tables in an SAP system are assigned to a data class, also known as TABART. During the export preparation, R3ldctl generates one structure file (STR file) for each data class. Each structure file is processed by a single R3load process. You can specify a size limit for the data packages that are generated by the R3load process in the corresponding *.cmd file, but all data packages of the structure file are created sequentially by one R3load process. Therefore, depending on the amount and size of tables defined in the structure file, the export can take a long time.

5.4.1 SAP data classes

Figure 5-8 on page 75 shows the standard data classes that are normally used in the SAP environment.

Usually there are many tables in the APPL-data classes, and typically, there are some large tables in data class APPL1.

When exporting such a system without splitting the structure files, the processing of packages that contain several large tables can significantly affect the total runtime of the export.

One solution is to split a single structure file into multiple files with the Package Splitting Tool.

Type	TABART	Usage
Data	APPL0	Master data, transparent tables
	APPL1	Transaction data, transparent tables (big tables)
	APPL2	Organization and customizing
	USER, USER1	Customer data class
Special	CLUST	Cluster tables
	POOL	Pool Tables
System	SAUS	Exchange tables for Upgrades
	SDIC	ABAP Dictionary tables
	SDOCU	Documentation
	SLDEF	Repository switch (SAP Upgrade)
	SLEXC	Repository switch (SAP Upgrade)
	SLOAD	Screen and report loads
	SPROT	Spool and logs
	SSDEF	Repository switch (SAP Upgrade)
	SSEXC	Repository switch (SAP Upgrade)
	SSRC	Source of screens and reports
BW	DDIM	Dimension tables
	DODS	ODS, PSA tables
	DFACT	Fact tables

Figure 5-8 Data classes (TABART)

5.4.2 Package Splitter tool

The Java STR Package Splitter tool provides the following features:

- ▶ Splits the largest <n> tables into separate structure files. Each file contains a single table.
- ▶ Splits tables that exceed a defined size limit into separate structure files. Each file contains a single table.
- ▶ Splits structure files for which the tables exceed a defined size limit. Each file might contain one or more tables.
- ▶ Splits tables into separate structure files by specifying their names in an input file. Each file contains a single table.

You can run the package splitter manually or by using SWPM. For details that describe how to use the package splitter tool, see the *SAP System Copy Guide*.

If you want to know how many tables belong to each data class in your system, you can use the SAP DBA Cockpit. In the DBA Cockpit, go to **Configuration** → **Data Classes**. In addition, the report SMIGR_CHECK_DB6 can be used to identify tables with a mismatch between the data class definition and the physical location.

Note: Historically, data classes are used to assign tables to tablespaces. Starting with *SAP SL Toolset 1.0 SP18*, tablespace pools were introduced with Db2. With tablespace pools, you can achieve a more even distribution of tables in your SAP system. For more information, see *SAP Note 2267446 - DB6: Support of tablespace pools*.

5.5 Local or remote export server scenarios

Exporting the system uses a large amount of CPU resources. The workload not only includes a database-related workload, but can also include CPU usage that is required for code page conversion, export dump file compression, and declustering.

Starting the export from one or more dedicated application servers is an option to release resources on the SAP database server and to shift the R3load related processing to different machines.

This can help if the single thread performance of the source database server is slower than that of a dedicated application server. Besides using dedicated application servers, it might help by using the importing database server for this purpose.

The following sections describe exports tests that were performed by using two different system setups:

1. Exporting from the SAP database server

The exporting R3load is started on the same machine where the Db2 database resides and thus competes with the database for system resources.

2. Exporting from an SAP application server

The exporting R3load is started on an SAP application server on a separate machine. The application server is remotely connected to a Db2 database. Export dump files are written to the application server machine. With this setup, R3load and Db2 use their own, distinct system resources.

5.5.1 Exporting from the SAP source database server

The exporting R3load runs on the same machine where the database resides that is being exported and thus competes with the database management system for resources.

Figure 5-9 shows the export duration of some tables including a Unicode conversion. It indicates that the runtime for all tables except CDCLS was around 1 hour, whereas the CDCLS export took nearly 8 hours.

package	time	size MB	Timeline (08-11 14:00 to 08-11 19:00)			
COEP	1:08:57	1033.29	[Progress bar]			
GLFUNCA	0:59:18	574.65	[Progress bar]			
BKPF	0:58:17	692.02	[Progress bar]			
BSIS	0:56:53	798.42	[Progress bar]			
GLPCA	0:53:14	422.71	[Progress bar]			
CDCLS	7:53:09	6921.85	[Progress bar]			
SWWLOGHIST	1:07:45	1605.92	[Progress bar]			
ACCTIT	1:00:16	689.09	[Progress bar]			

Figure 5-9 Table export run times using database server

The NMON utility on AIX can show the CPU consumed by Workload Management Classes. This can be used to monitor CPU usage of different users and processes. Monitor the CPU usage by R3load that is started as the SAP administrator <sid>adm and the Db2 processes and threads that are running under the Db2 instance owner db2<sid>.

Figure 5-10 on page 77 shows the CPU resource consumption. This illustrates that R3load is using most of the available CPU resources, whereas Db2 is using only a minor part.

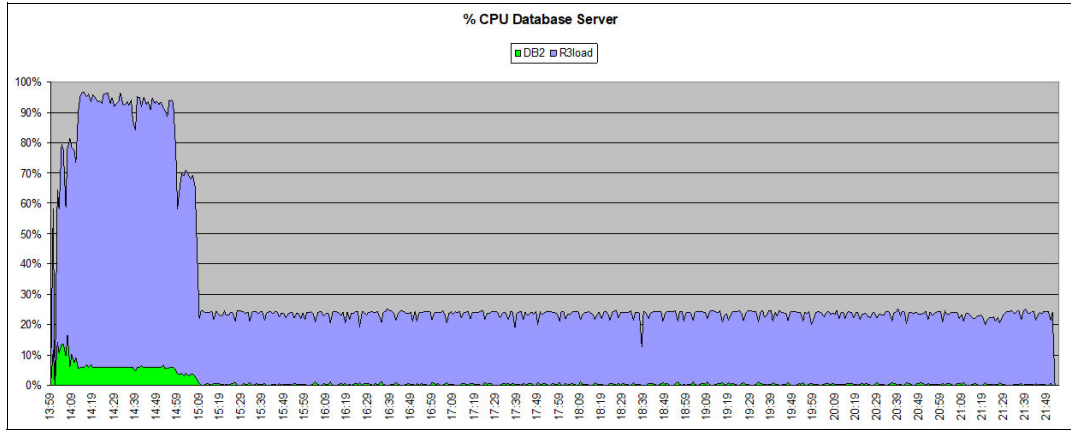


Figure 5-10 CPU Resource Consumption - SAP Database Server

Figure 5-10 shows that at the beginning of the export when 8 R3load processes were running in parallel, the CPU usage was nearly 100%. If the workload generated by R3load is moved to another machine, this frees CPU resources on the DB Server.

5.5.2 Exporting from an SAP application server

Figure 5-11 shows export runtimes when using an SAP application server to perform the export. These are the same tables shown in Figure 5-9 on page 76. Although the runtime decreased for all tables, the CDCLS table runtime is reduced by approximately 45%.

package	time	size MB	08-14 16:30	08-14 17:30	08-14 18:30	08-14 19:30
CDCLS	4:35:34	6921.39	[Progress bar]			
COEP	0:49:31	1033.29	[Progress bar]			
GLFUNCA	0:48:01	574.65	[Progress bar]			
ACCTIT	0:46:28	689.09	[Progress bar]			
SWVLOGHIST	0:46:27	1605.92	[Progress bar]			
BSIS	0:45:13	798.42	[Progress bar]			
BKPF	0:43:33	692.02	[Progress bar]			
GLPCA	0:40:27	422.71	[Progress bar]			

Figure 5-11 Table export runtimes using SAP application server

Figure 5-12 and Figure 5-13 on page 78 show the CPU usage for both the SAP database server and for the dedicated application server.

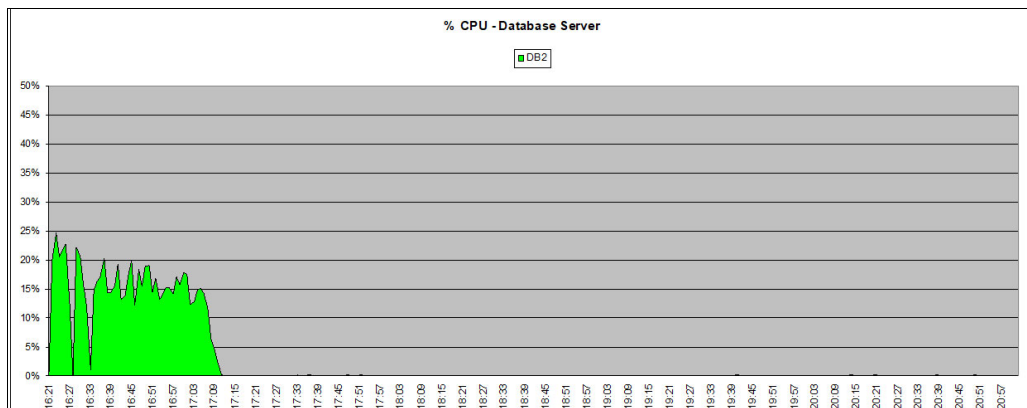


Figure 5-12 CPU Resource Consumption - SAP Database Server

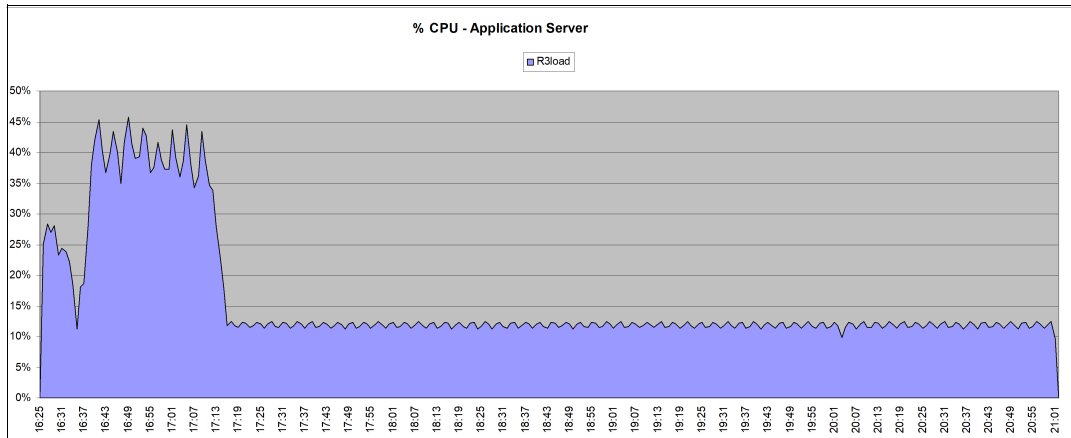


Figure 5-13 CPU Resource Consumption - SAP Application Server

When you compare the CPU usage on the database server to the previous test performed, the Db2 database engine can use more CPU resources because the CPU load that was generated by R3load was offloaded to the application server.

5.5.3 Identifying candidates for dedicated application servers

It is a best practice to closely monitor the resource usage. A monitoring setup with different workload classes can be complex or not even possible. You can use an alternative source of information to analyze the different sources of workload.

The R3load log files contain a resource usage section at the end of each file, which is shown in Example 5-6. This section can be used to identify candidates to move to dedicated application servers. This resource usage section distinguishes between the time that is spent in the database, the general R3load processing, and the file processing. One example might be if the time spent in the GENERAL section is significantly higher compared to the DATABASE section.

Note: The resource usage for the R3load compression is reported under the FILE section in this report. A large amount of time in the FILE section does not necessarily indicate an I/O bottleneck.

Example 5-6 Excerpt of the R3load log file with resource usage data

GENERAL	times:	3998.905/1944.467/0.020	real/usr/sys
DATABASE	times:	387.097/ 35.725/1.062	real/usr/sys
FILE	times:	1367.156/ 587.576/15.46	real/usr/sys

5.5.4 Recommendation for dedicated application servers

The usage of dedicated application servers for the export can increase network usage as more data between the database server and the application server is transferred. The R3load export dump files are compressed, but the data flowing between the database server and the client machine is not compressed by Db2. Therefore, the total amount of data that uses the network increases. For table clusters, this might be less significant as the data cannot be compressed well. Shifting the processing of table clusters to dedicated application servers might have only a limited effect on network usage.

In general, the use of dedicated application servers competes with the data transfer of other export dump files to the target server. This might mean that the process might not be beneficial to the overall migration process. If you want to use this optimization, ensure that you closely monitor the resource usage for CPU and network and shift only a portion of the export to dedicated application servers.

In the example discussed in section 5.5.2, “Exporting from an SAP application server” on page 77, a good setup is to use a dedicated application server only for the CDCLS table as it determines the overall runtime of the migration process.

The use of a dedicated application server for only selected tables cannot be done by using SWPM alone. Instead, you must have two separate instances of the Migration Monitor with each running with a mutually exclusive set of tables. Before you use this feature, become familiar with the Migration Monitor and how to use it outside the SWPM.

5.6 Db2 query block prefetch feature

For IBM Db2 11.5.6 and later, the Db2 CLI client provides a *query block prefetch* feature. This new feature helps to reduce the effect of network latencies for SQL queries with a large result set.

5.6.1 Feature description

By default, data is sent to and from the database management system to the Db2 client in blocks, fetched by the client application. After all data is received and processed for a request, a new request for data is sent to the database engine. After that request is received by the Db2 engine, data is prepared and sent to the client. This process is repeated until the complete result set is transferred to the client.

With the query block prefetch feature enabled, the Db2 client requests another block before the client application processes all records.

5.6.2 Performance impact

As shown in Figure 5-14, this feature can optimize the export processing on the client and the Db2 database engine. Sometimes, this feature can save 50% of the elapsed time for a large result set.

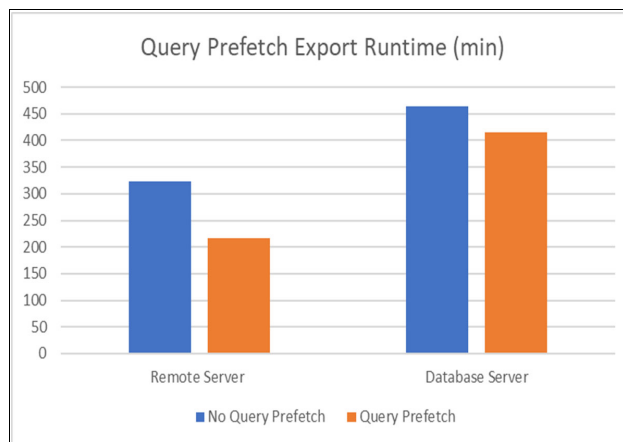


Figure 5-14 Query Prefetch runtime comparison

Based on the tests conducted for this book, the export throughput increases to a maximum of 33% if the export is done on a dedicated application server. The export throughput increases by 10% if the export is started on the database server itself.

The test results shown in Figure 5-14 on page 79 were obtained by running 6 parallel R3load processes and the network bandwidth was not saturated. Adding more processes that saturate the network bandwidth can reduce or possibly eliminate the advantage provided by the query prefetch feature.

This feature however, matches to the previously proposed setup for dedicated applications servers during the export. This setup is most valuable if only a few critical tables are exported on dedicated applications servers.

Even if the export is started on the database server, the feature can help because the local Db2 client also communicates by using the network protocol, and the same optimizations apply. The benefit is not as significant as seen with a physical network, but the test still showed a 10% improvement.

5.6.3 Enabling Db2 query block prefetch

To enable CLI query block prefetch, you add the key QUERY_PREFETCH=1 to your `db2cli.ini` file, which is located in `/sapmnt/<SID>/global/db6` as shown in Example 5-7.

Example 5-7 A sample db2cli.ini file with Query Prefetch enabled.

```
[SID]
Database=SID
Protocol=tcPIP
Hostname=mydbhost
Servicename=5912
QUERY_PREFETCH=1
[COMMON]
Diagpath=/usr/sap/SID/SYS/global/db6/db2dump
```

If you are running on Db2 Version 11.5.6 or later, enable this feature for both local and remote exports. If you are running on a Db2 version earlier than 11.5.6 on the export server, you can use the Db2 version 11.5.6 or later client for testing. As the query block prefetch feature is a client enhancement, this can be used also with earlier Db2 releases.

The use of different client and server versions generates an error and the connection is refused. If the R3load generates an error `ERROR => Db2 software level mismatch between client and server`, follow the instructions provided by *SAP Note 503122 - DB6: Connection refused due to DB software mismatch* to overcome this.

Note: A mixed version setup is only recommended for testing purposes. If you find the feature useful, upgrade to Db2 Version 11.5.6 or later as part of the migration process.

5.7 Db2 workload management

You can use Db2 workload management (WLM) to distinguish and prioritize different types of work on the database. With SAP Systems, Db2 WLM is mainly used to monitor resource consumption of the different workloads but not to restrict workloads from being run.

However, when you use Db2 columnar tables, also called BLU tables, a threshold for the maximum number of concurrent queries is recommended and enforced during installation of a Db2 fix pack update by the `db6_update_db` script.

For more information, see the database administration guides for *Db2 and SAP Note 1152411 - DB6: Using Db2's Workload Management in an SAP Environment*.

Db2 WLM threshold impact on export

Although a Db2 threshold for the number of BLU queries that can run in parallel in the database is valid for normal operation, it also limits the number of parallel R3load processes.

For example, if the source system has 128 cores but the Db2 WLM threshold is set to 16, only 16 parallel R3load processes are concurrently active, regardless of whether they export columnar oriented or row-based tables. The blocked R3load processes do not fail immediately but wait in a queue to be run.

So, for heterogeneous system copies, disable the WLM threshold, or set it to a high number. You can use the SAP report `SMIGR_CHECK_DB6` or the Db2 SQL query shown in Example 5-8 to see whether a threshold is set and enabled.

Example 5-8 SQL Statement to check status of Db2 WLM threshold

```
SELECT SUBSTR(THRESHOLDNAME,1,20) AS NAME, MAXVALUE, ENABLED FROM
SYSCAT.THRESHOLDS
```

NAME	MAXVALUE	ENABLED
-----	-----	-----
SYSDEFAULTCONCURRENT	60	Y

After you have verified that the Db2 WLM threshold is enabled, you can either increase the threshold to the maximum number of parallel R3load processes or you can disable the threshold using the SQL statements shown in Example 5-9 and Example 5-10.

Example 5-9 SQL Statement to increase the parallelism of the Db2 WLM threshold

```
ALTER THRESHOLD <THRESHOLDNAME> WHEN CONCURRENTDBCOORDACTIVITIES > <NEM_MAXVALUE>
STOP EXECUTION
```

Example 5-10 SQL Statement to disable and drop Db2 WLM threshold

```
ALTER THRESHOLD <THRESHOLDNAME> DISABLE
DROP THRESHOLD <THRESHOLDNAME>
```

Note: If you run a test export on the production system, be sure to switch the configuration back to its initial state. If you run a test export on a system copy, be sure to disable the Db2 WLM threshold again for the final migration.

5.8 Other Db2 features

Looking at the process and resources used during the export, there might be other potential areas of export optimizations. In tests and during projects with customers, it was determined that the Db2 configuration used during normal production also works well during the export. There are some areas that might improve the export, but the benefit depends on the system environment.

In some environments, the following optimizations might provide some additional throughput during export:

- ▶ SMP parallelism

The Db2 Registry Variable `INTRA_PARALLEL=YES` might have a positive performance impact if the system has available CPU and I/O resources. Most likely, this is because that CPU and I/O resources are typically the most used resources in the system. If you want to try this feature, refer to *SAP Note 2047006 - DB6: Use of Db2 SMP Parallelism (INTRA_PARALLEL=YES)* for details.

- ▶ Increased sort area

If sort overflow occurs during a sorted export, Db2 might write data to disk in temporary tablespaces. This generates I/O and requires additional CPU cycles because of bufferpool writes and reads. If the export of a sorted table determines the overall downtime, it can be beneficial to increase the Db2 database parameters `SORTHEAP` and `SHEAPTHRES_SHR` and reduce the size of the bufferpool. If the increases to `SORTHEAP` and `SHEAPTHRES_SHR` reduces the writes to disk, then the I/O and CPU usage decreases. In our tests, it was possible to avoid some writes to disk by using a smaller bufferpool at the cost of an overall performance degradation during the export. Choose this option judiciously and include detailed Db2 monitoring to ensure optimal results.

- ▶ Self-Tuning Memory Manager (STMM)

During normal operation, Db2 STMM should be active to manage the memory areas for Db2. It can also be active during test migrations and can also be adapted to optimize the export. A best practice for the final migration is to use the same configuration that was used for the last successful test migration. During the final system copy, consider using the same fixed values that were optimized by Db2 STMM.

- ▶ Additional indexes

For unsorted exports, there is no benefit of an additional index as all data is accessed and exported. Db2 often uses a table scan to get all data. For sorted exports, indexes may be beneficial.

You can create a compound index that includes all columns and which provides index only accesses. We have not seen a benefit of using additional indexes during our migration projects. Therefore, this optimization might be used infrequently. An additional index is most likely to help for large tables only. The additional indexes must be created during normal operation before the migration. The creation of the additional indexes increases the database footprint and might negatively impact insert performance during normal operation. To implement a meaningful index, you must analyze the SQL statement of the R3load process and use the Db2 explain function and the Db2 index advisor to find an appropriate index. For details, refer to *SAP Note 2305865 - DB6: Optimizing Performance with Indexes*.

- ▶ Table Reorganization

If data is fragmented on disk, consider reorganizing a table before the export. However, as with additional indexes, reorganization was not used in recent projects.

You might choose this option if the export of one table determines the overall downtime of the migration and if all other optimizations are already implemented.

To perform the reorganization, refer to *SAP Note 1942183 - DB6: When to consider a table or index reorganization*. Along with the information in that SAP note, analyze the table with the DBA Cockpit and check whether there are many overflow accesses. Alternatively, you can use the `MON_GET_TABLE` table function shown in Example 5-11 on page 83 during or after the export and check for a high value of overflow accesses.

Example 5-11 SQL Statement to get overflow accesses for table BALDAT

```
SELECT varchar(tabschema,20) AS SCHEMA, varchar(tabname,20) AS NAME,  
ROWS_READ, OVERFLOW_ACCESSES FROM TABLE(MON_GET_TABLE('BALDAT',-2))  
as T
```



Basic Db2 layout and configuration options

This chapter briefly describes some basic Db2 recommendations regarding tablespace layout and Db2 configuration.

The following topics are discussed in this chapter:

- ▶ 6.1, “Planning your Db2 target system” on page 86
- ▶ 6.2, “Tablespace principles and configuration” on page 86
- ▶ 6.3, “Tablespace pools” on page 87Figure 6-1 on page 87
- ▶ 6.4, “General layout considerations” on page 91
- ▶ 6.5, “R3LOAD with Db2 compression” on page 96
- ▶ 6.6, “Import optimization” on page 106
- ▶ 6.7, “Using and optimizing Db2 LOAD with R3load” on page 110
- ▶ 6.8, “Order and configuration of index creation” on page 117
- ▶ 6.9, “Db2 buffer pools” on page 120
- ▶ 6.10, “SMP parallelism for index creation” on page 121
- ▶ 6.11, “LOCKTIMEOUT” on page 123
- ▶ 6.12, “Optimizing statistics collection” on page 123
- ▶ 6.13, “Importing by use of a database server versus a remote client” on page 126
- ▶ 6.14, “Other Db2 features” on page 127

6.1 Planning your Db2 target system

Before you start the import into the Db2 target system, define a plan for the database layout and database configuration.

Plan the disk assignment together with your cloud service provider as different providers have different types of disks and different concepts of how required throughput, IOPS, and size can be assigned to the virtual machines.

The heterogeneous system copy also includes the possibility to assign tables and indexes to dedicated tablespaces. With this, you can optimize the layout with respect to manageability and performance.

6.2 Tablespace principles and configuration

Db2 supports multiple concepts to store data in tablespaces. The two different basic concepts are the System Managed Space (SMS) tablespaces and the Database Managed Space (DMS) tablespaces. The enhancement to these concepts is called automatic storage and is the default for Db2 tablespaces.

During the migration, the database size and the size of tablespaces and containers are defined in the file DBSIZE.XML that is created during the export. The SAP installation tool SWPM provides several options to choose from. For example, by using the storage path option you can specify the number of file systems (sapdata1 – sapdata<n>) used for the Db2 database. With the tool, you can also create tablespaces manually.

The number of containers has an impact on performance, and you cannot easily change the configuration while the system is running in production. A discussion about the appropriate disk layout is beyond the scope of this book and also depends on the underlying disk subsystem.

However, some basic recommendations remain the same for all implementations:

- ▶ Use separate disks for logging and for tablespaces.
- ▶ Do not configure operating system I/O, such as swap, paging, or heavily used spool, on Db2 data disks or disks that are used for logging.

With the configuration parameters described in this chapter, the appropriate layout and data placement is essential for the optimal operation of the database. You can also assign large tables to dedicated tablespaces to optimize the layout. As the growth of the SAP database is monitored in production, use the rate of growth to estimate the future size of the database.

Figure 6-1 on page 87 shows a sample list of the largest tablespaces in an SAP database. We use this example to show what a tablespace layout should *not* look like.

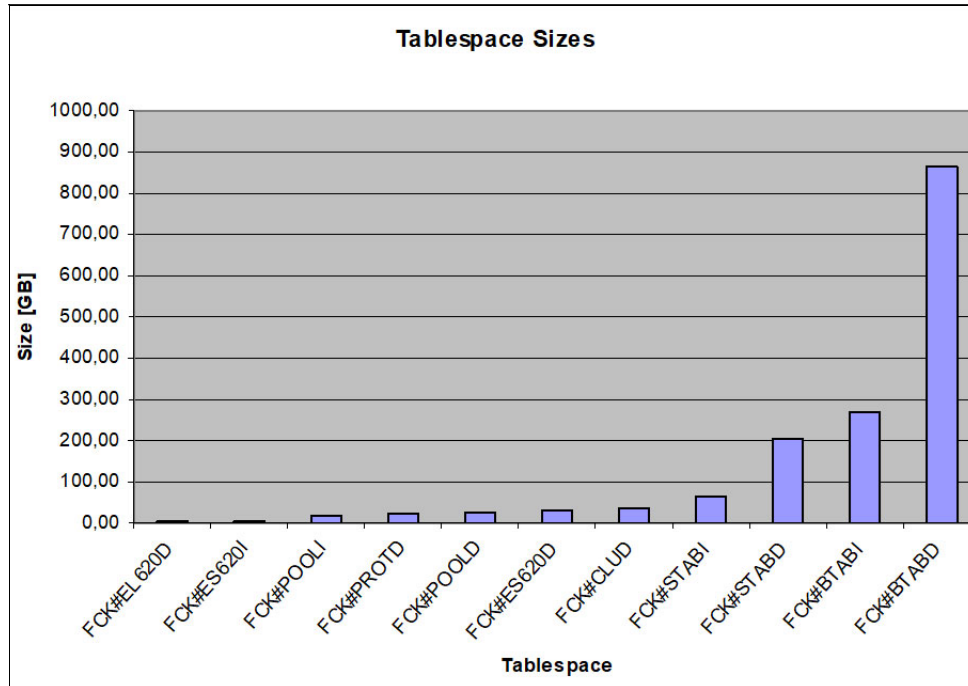


Figure 6-1 Tablespace Sizes (Not Optimized)

In this case, the BTABD tablespace is larger by than the next smaller tablespace orders of magnitude. This might not have an impact on the performance for the daily workload, but it does have an impact on administration. For example, when backing up your database Db2 is backing up tablespaces in parallel to improve performance. This improvement cannot be fully used if one tablespace is factorially larger than the others.

To optimize the layout, the largest tables can be assigned to dedicated tablespaces. Tablespace pools can be used, or both options can be combined.

6.3 Tablespace pools

In an SAP system, which uses a traditional tablespace layout, you typically find 20 to 60 tablespaces. Each table is stored within one of these tablespaces. Tables can be grouped logically and assigned to dedicated tablespaces.

With the traditional Db2 tablespace layout for SAP systems, this type of functional grouping is done by grouping master data and transaction data, for example. This layout typically results in a few large tablespaces that contain one or more large tables. However, this kind of unbalanced tablespace setup might result in some issues, such as increased backup run times.

The concept of tablespace pools describes a more even distribution of tables to tablespaces. In addition, the tablespace pool concept establishes a separation of regular, index, long field (LF), and large object (LOB) data into dedicated tablespaces. In comparison, a traditional layout typically does not separate LF and LOB data from regular data.

Therefore, the tablespace pool layout comes with several advantages:

- ▶ **Optimized backup and restore run times**
 Database backup and restore operations are parallelized at the tablespace level. This means, if your system has an unbalanced tablespace layout, with a few large and many smaller tablespaces, then parallelizing the backup process will not work in the most optimal way and the largest tablespaces determine the overall backup runtime. Tablespace pools typically show a uniform distribution of data among tablespaces. By using tablespace pools, backup and restore operations can run with an ideal degree of parallelism, and the runtime is expected to improve significantly.
- ▶ **Better performance optimization:**
 Use the separation of regular table data from LF and LOB to more finely tune performance optimization. LF and LOB data is not cached in the Db2 buffer pool but is read directly. By storing this type of data into dedicated tablespaces, you can selectively enable file system caching for the respective tablespaces to allow buffering at the file system level. This type of caching is enabled by default for the long tablespaces, when using tablespace pools.
- ▶ **Minimize risk of hitting maximum table objects:**
 There is a limit on the maximum number of table objects that can be stored in a single DMS tablespace. When using 16k pages, this limit is 53,747. With an unbalanced tablespace layout, there is a significantly higher chance to reach this limit. For example, this might happen during an SAP upgrade. The use of tablespace pools minimizes this risk.

To summarize, we recommend using tablespace pools. Building a new target system as part of a system copy process provides an opportunity to implement the tablespace pool concept.

6.3.1 Prerequisites and functionality of tablespace pools

The main prerequisite for using tablespace pools is automatic storage tablespaces. If your source system does not use automatic storage yet, we recommend that you switch to automatic storage in your target system. If the other prerequisites, which are listed in SAP note “2267446 - DB6: Support of tablespace pools” are met, such as an up-to-date version of the database shared library, then tablespace pools can be easily implemented as part of your system copy process. SAP SL toolset 1.0 SP18 or higher supports tablespace pools.

In contrast to the traditional tablespace layout, where SAP data classes are assigned with dedicated tablespaces, data classes are assigned to a tablespace pool. Table 6-1 shows an example.

Table 6-1 Tablespace pool example

SAP Data Class	Traditional tablespace layout	Tablespace Pool(s)
APPL0	<SAPSID>#STABD <SAPSID>#STABI	<SAPSID>#DATA{20}
APPL1	<SAPSID>#BTABD <SAPSID>#BTABI	<SAPSID>#DATA{20}
APPL2	<SAPSID>#POOLD <SAPSID>#POOLI	<SAPSID>#DATA{20}
CLUST	<SAPSID>#CLUD <SAPSID>#CLUI	<SAPSID>#DATA{20}

The assignment of data classes to tablespaces is stored in the following tables: TADB6 (for tables), IADB6 (for indexes) and LADB6 (for long field and large object data). Note that the LADB6 table based definition is only fully used when using tablespace pools.

A tablespace pool follows a specific naming convention:

- ▶ A prefix: <SAPSID>, followed by the “#” character
- ▶ Name of the pool, which defaults to “DATA”. The default can be modified as part of the installation process.
- ▶ Size of the pool in curly brackets {}. The default pool size is 20. Its minimum size is 10 and the maximum size is 99.

A tablespace pool consists of multiple pool tablespaces. Depending on the pool size, an equal number of tablespaces for regular data (D), index (I) and large tablespaces (L) is created.

Assume that we create a tablespace pool named “DATA” with a size of 20. Then, the following 60 pool tablespaces are created as shown in Table 6-2.

The naming scheme for pool tablespaces is as follows:

- ▶ A prefix: <SAPSID>, followed by the “#” character
- ▶ Name of the pool, which defaults to “DATA”
- ▶ Number of the pool tablespaces after the “@” sign. This is an increasing number between 01 and 99
- ▶ Type of tablespace: D for regular data, I for indexes, L for long data tablespaces (containing LF/LOB data).

Table 6-2 Tablespaces in a tablespace pool

<SAPSID>#DATA@01D	<SAPSID>#DATA@01I	<SAPSID>#DATA@01L
<SAPSID>#DATA@02D	<SAPSID>#DATA@02I	<SAPSID>#DATA@02L
<SAPSID>#DATA@03D	<SAPSID>#DATA@03I	<SAPSID>#DATA@03L
...
<SAPSID>#DATA@20D	<SAPSID>#DATA@20I	<SAPSID>#DATA@20L

When using a tablespace pool, the distribution of tables, indexes, and long field or large object data to tablespaces no longer follows a predefined, static definition from the SAP data dictionary. Instead, objects are mapped to tablespaces dynamically and automatically using a hash algorithm. This hash functionality is built into the DB6 database shared library (DBSL). The input value for the hash algorithm is the table name and its output is the number of the target pool tablespace. For instance, if the algorithm hashes the name of table “BSIS” to the value “05”, then this table is created in tablespace <SAPSID>#DATA@05D and its indexes in tablespace <SAPSID>#DATA@05I.

By using the hash algorithm, an equal distribution of tables across the available pool tablespaces can be achieved. You can use SAP report “RSDB6TBSPOLDISTRIB” to simulate the distribution of objects to pool tablespaces.

Note: The pool does not have to exist when running the simulation, so you can run this report in your source system prior to the database export.

6.3.2 Usage and recommendations of tablespace pools

An R3load-based system copy provides an opportunity to optimize your tablespace layout for your target system. This is especially true for establishing the tablespace pool concept. We recommend that you use tablespace pools in your target system.

For SL Toolset 1.0 SP18 or later, the Software Provisioning Manager (SWPM) allows you to set up tablespace pools, both for new installations and during an R3load based system copy. If your source system already runs with tablespace pools, the configuration is preserved in your target system. If your source is based on the traditional layout, you can use SWPM to define tablespace pools.

Note: For SL Toolset 1.0 SP18 or later, the use of tablespace pools is the default in SWPM. If you want to preserve the traditional layout, choose *custom mode* in SWPM and clear the **use tablespace pools** checkbox.

For small to mid-sized systems, you can decide to store all objects in a tablespace pool. However, for larger systems, especially if they contain a few large tables, we recommend associating these large objects with their own, dedicated tablespaces.

If you already have performed a separation of large tables in your source system and used custom data classes for these objects, the no further configuration is necessary. The use of tablespace pools in your target system only affects SAP's standard data classes, that is: APPL0, APPL1, APPL2, CLUST, POOL, SDIC, SDOCU, SLDEF, SLEXC, SLOAD, SPROT, SSDEF, SSEX, SSRC, TEMP, USER, and USER1. Therefore, all objects associated with custom data classes, such as prefixes Z, Y, or USR and USER2 to USER9, or with BW specific data classes, such as DFACT, DDIM, and DODS are not assigned to the tablespace pool but stay in their original tablespaces.

If you are running a mid-sized to large system with a few large tables that are not residing in their own tablespaces yet, consider assigning custom data classes to those tables in your source system before export. If you assign customer data classes, the tables are assigned to their own, dedicated tablespaces in the target system. You can assign data classes with the following steps:

1. Create the new dedicated tablespaces in your source system with minimal size by using the DBA Cockpit using **Space** → **Tablespaces** → **Add**.
2. In DBA Cockpit, create a new data class and assign the newly created tablespaces to it using **Configuration** → **Data Classes**. Check *SAP note "515968 - DB6: Creating data classes and tablespaces in DBA cockpit"* for details.
3. In transaction SE11, go to the technical settings for the respective tables and modify the data class
4. Do *not* perform the table conversion now. The tables are moved into the new target tablespaces when you perform the import into the target system.

Note: The migration check report SMIGR_CHECK_DB6 shows this as an inconsistency in tablespace assignment. However, you may ignore it, as it is intended in our case.

Alternatively: If you perform the assignment of new custom data classes before the migration, then consider performing the table conversion in the source system as described in *SAP note "1513862 - DB6: Table conversion using DB6CONV version 6 or higher"*.

The new assignment of tables to tablespaces and data classes takes effect during the migration, and you can validate assignments by checking the DDLDB6.TPL File.

Example 6-1 shows a classic layout and Example 6-2 shows the tablespace pool layout.

Example 6-1 Excerpt from DDLDB6.TPL with the classic layout and custom dataclass ZAPP1

```
# table storage parameters
loc:  USER6 NZW#ES40AD          NZW#ES40AD          NZW#ES40AD
      APPL1 NZW#BTABD          NZW#BTABI           NZW#BTABD
      ZAPP1 NZW#BTAB1D         NZW#BTAB1I         NZW#BTAB1D
```

Example 6-2 Excerpt of DDLDB6.TPL with tablespace pools and custom dataclass ZAPP1

```
# table storage parameters

loc:  APPL0 SRC#DATA{20}        SRC#DATA{20}        SRC#DATA{20}
      APPL1 SRC#DATA{20}        SRC#DATA{20}        SRC#DATA{20}
      APPL2 SRC#DATA{20}        SRC#DATA{20}        SRC#DATA{20}
      ZAPP1 SRC#ZBSISD          SRC#ZBSISI           SRC#TBSISD
```

To evaluate the correctness of the tablespaces assignment, use the SAP report SMIGR_CHECK_DB6.

6.4 General layout considerations

This section provides several general recommendations about how you configure your Db2 environment.

6.4.1 File system caching

If file system caching is enabled, the operating system attempts to minimize the frequency of disk access. This is done in main memory within a structure called the file system buffer cache. With many applications this leads to increased performance. However, certain classes of applications derive no benefit from the file system buffer cache.

With Db2, the default I/O mechanism for newly created tablespace containers on most UNIX, Linux, and Windows platforms is concurrent I/O (CIO) or direct I/O (DIO) and should be used during the heterogeneous system copy and for normal production also.

6.4.2 Page size

Tablespaces are created based on a certain page size. Four supported page sizes exist: 4 KB, 8 KB, 16 KB, and 32 KB. SAP recommends that you use a page size of 16 KB. The migration should be performed with this recommendation. Our tests and customer experiences show that there is no impact on performance with a different page size during the migration.

6.4.3 Extent size

Workload tests and numbers collected from customer systems have shown that I/O performance is not sensitive to the extent sizes in production workloads. This is primarily because modern storage subsystems employ sequential detection and that prefetching can be influenced by the prefetch size parameter for Db2 tablespaces.

However, the extent size influences the size of the database because it is the minimum allocation unit for tables. A large extent size might result in wasted disk space for small or empty tables.

The current SAP recommendation is to use an extent size of 2, which together with the recommendation of 16 KB pages is 32 KB. This is a good default for the tablespaces, including Db2 system temporary tablespaces.

You can use a larger extent size if wasting space because of small tables is not an issue. Larger extent sizes have other benefits such as improved backup performance. The extent size can also have an influence on the import. We have seen a single digit increase in throughput with extent size of 16. With larger extent sizes, we have seen a slight decrease in performance.

You can create tablespaces with extent size of 16 for large tables that reside in a dedicated tablespace or for dedicated tablespaces that serve the temporary tables during a split table load.

Note: SAP SWPM always creates the tablespaces with extent size 2. If you want to use a different extent size, you must create the tablespaces manually and grant the right to use the tablespace to the role SAPAPP.

6.4.4 NUM_IOCLEANERS

You can use the NUM_IOCLEANERS parameter to specify the number of asynchronous page cleaners for a Db2 database. Page cleaners write changed pages from the buffer pool to disk before a database agent requires space in the buffer pool.

Because we use the db2load API to load the biggest tables, this parameter is not that important for the overall migration process. However, for index creation, if temporary data is written from buffer pool to temporary tablespace, this has some impact on performance.

During the migration evaluation project, tests showed no additional improvement by defining more than 20 I/O cleaners for index creation. Unless the disk throughput is below the expected performance for a given I/O configuration, we recommend that you set this configuration parameter to **AUTOMATIC** and let Db2 determine the appropriate number.

6.4.5 Prefetch size

The tablespace prefetch size determines the degree to which separate containers can operate in parallel and how much data is read ahead when prefetching is initiated by the database. Typically, prefetching occurs during the index build phase within the db2load API or in a dedicated create index statement. Index builds benefit from a large prefetch size.

Set the prefetch size to **AUTOMATIC** as a starting point and change it only if the I/O performance is below the expected rate for the given disk subsystem.

6.4.6 Db2 parallel I/O

Db2 assumes that each container has one physical disk that is assigned on the underlying storage. Because the SAP default is to use at least 4 storage paths for data and an additional 4 storage paths for temporary tablespaces, I/O parallelism is already ensured.

To increase the parallelism even further, you can set the Db2 registry variable `DB2_PARALLEL_IO=NUMBER`. With this, Db2 uses a parallelism of *NUMBER* for each Db2 Container.

The recommendation of 4 Storage paths is intended to be the minimum or a feasible value for smaller systems. With larger databases, define a maximum of 16 storage paths and control parallelism using this.

We do not see any performance improvement, when setting this variable. Therefore, the recommendation is to leave it as the default.

Note: Contact the cloud service provider for details about the best storage layout and logical volume manager (LVM) configurations. The I/O sizing is critical for the migration and the subsequent production operation.

6.4.7 Support for 4K sector size

Modern disks have a sector size of 4K. This is also true for storage provided in a cloud environment. For example, Azure Ultra Disks support this sector size.

Db2 11.5 and higher supports this modern disk format and can optimize the I/O to type of disk with the registry variable `DB2_4K_DEVICE_SUPPORT=ON`. If this variable is set to ON, this setting adjusts the memory layout of data structures and the parameter of disk I/O operations to be compatible with the requirements of this type of storage.

To create a database on disks with a 4K sector size, this registry variable needs to be set to ON. Otherwise, the created database statement fails.

If you enable this setting in an environment configured with storage devices that use a 512-byte sector size, performance might be degraded. Disks with 4K sector support can also provide a 512-byte compatibility mode.

Setting `DB2_4K_DEVICE_SUPPORT=ON` for 4K Disks in 512-Byte compatibility mode is not recommended. However, we have not tested the impact of this registry variable, so we cannot assess the impact during the heterogeneous system copy.

6.4.8 Preallocation of tablespaces

With Db2 automatic storage, tablespace containers automatically increase in size if enough space is available in the underlying file system. If a tablespace becomes full, Db2 increases the tablespace automatically. You can influence the automatic setting by setting the Db2 configuration parameter `INCREASESIZE` to a fixed value or a percentage of the existing size.

During the import, tablespaces with an initial size increase, and therefore, it is an option to manually set the increase size to a large value, such as 1 GB. Db2 enables fast preallocation of containers on VxFS, JFS2, GPFS, ext4 (Linux only) and xFS (Linux only) file systems by default and so the duration for the increase is short but still exists.

Also it is also a best practice after a first test migration to analyze the tablespace sizes and create the tablespace manually to move this effort outside the downtime window completely.

You can extract the tablespace definition after the first test migration with the `db2look` utility. This utility can define the DDL Statements to create the tablespaces.

To extract the information, use the command:

```
db2look -d <DATABASE_NAME> -l
```

An example output is shown in Example 6-3.

Example 6-3 DDL Statement extracted after a test migration with the db2look utility

```
CREATE LARGE TABLESPACE "SRC#DATA@19D" IN DATABASE PARTITION GROUP SAPNODEGRP_SRC
    PAGESIZE 16384 MANAGED BY AUTOMATIC STORAGE
    USING STOGROUP "IBMSTOGROUP"
    AUTORESIZE YES
    INITIALSIZE 32 M
    MAXSIZE NONE
    EXTENTSIZE 2
    PREFETCHSIZE AUTOMATIC
    BUFFERPOOL "IBMDEFAULTBP"
    DATA TAG INHERIT
    OVERHEAD INHERIT
    TRANSFERRATE INHERIT
    DROPPED TABLE RECOVERY OFF;
```

In the second step, extract the current size of the tablespaces using the `MON_GET_TABLESPACE` table function and change the `INITIALSIZE` parameter for each tablespace.

Before the next migration run, you can create the tablespaces with the known size. You can combine this feature with setting the Db2 registry variable `DB2_USE_FAST_PREALLOCATION=OFF` to minimize fragmentation on file system level as much as possible.

This will avoid many small wait situations that occur in the time between when a tablespace full event occurs and when the increase is completed. The wait is typically only a few seconds, but we have seen a 5% increase in throughput with this concept.

Important: Do not forget to reset the modified `INCREASESIZE` definition again after the migration completes.

6.4.9 Logging configuration

The migration process is a process of moving data from one database to another using the SAP utilities. So, database logging might be a bottleneck during the migration. If you are using the Db2 LOAD functionality through the db2load API, logging is reduced to a minimum. Therefore, the logging configuration while loading is not as critical as during production use of the database.

Db2 has two different modes of using log files:

- ▶ Circular logging mode
- ▶ Archival logging mode.

The archival logging allows point in time recovery. The circular logging mode allows only offline backups and no roll forward after a restore because log files are not archived but overwritten.

Database load is a special operation and a recovery situation during the migration almost always means failure of the migration and a restart. There is no need to configure archival

logging. In addition, the circular logging can be beneficial for performance as it does not log LOB data. Therefore, the amount of logging I/O will be reduced.

We recommend that you run the database in circular logging mode by setting the configuration parameters as shown in Example 6-4.

Example 6-4 Database Configuration for Circular Logging

```
First log archive method(LOGARCHMETH1) = OFF
Options for logarchmeth1(LOGARCHOPT1) =
Second log archive method(LOGARCHMETH2) = OFF
Options for logarchmeth2(LOGARCHOPT2) =
```

The Db2 database configuration variable LOGINDEXBUILD enables the database manager to log the creation of indexes and generates a significant amount of logging. By default, this variable is turned off. Verify this is true for your environment.

However, a certain amount of logging occurs during the migration because some tables are imported by INSERT, and the db2load API logs the allocation of new extents. Although the amount of logging is small compared to the production use, you can influence the behavior with the following configuration parameters:

- ▶ SOFTMAX
- ▶ LOGBUFSZ
- ▶ PAGE_AGE_TRGT_MCR
- ▶ The registry variable DB2_LOGGER_NON_BUFFERED_IO
- ▶ The registry variable DB2_USE_FAST_LOG_PREALLOCATION

SAP provides a good setting for Db2 transactional logging. You can leave the defaults.

Although logging is limited during a heterogeneous system copy, ensure that there is enough disk space available for the Db2 transactional log files. Also, ensure that the size and number is configured large enough by setting the following Db2 configuration parameters:

- ▶ LOGFILSIZ
- ▶ LOGPRIMARY
- ▶ LOGSECOND

Provide enough disk space and an appropriate number of correctly sized primary and secondary log files, and enable circular logging to optimize the migration in this area.

An exception might be if you use multiple parallel inserts for split table imports. In this case, closely monitor the logging performance. If the logging performance slows the migration, then increase the IOPS available to Db2 logging.

Note: After the system copy completes, review and set the logging parameters according to your system and adapt the configuration.

6.5 R3LOAD with Db2 compression

This section explains how data can be compressed during import into the IBM Db2 database.

6.5.1 Introduction to Db2 compression

Compression reduces the footprint of stored data for tables, indexes, and backup images. Compressing data is a well-established technique in Db2 and has been used in many systems for many years.

Apart from the space savings and the related cost savings, we often see additional performance benefits with using compression. When pages residing in the Db2 buffer pools that are stored in compressed format, more data fits into the given buffer pools as the same amount of data can be stored in fewer pages. This reduces the number of I/O operations and because many systems have more I/O limitations than CPU limitations, data compression typically leads to performance benefits. Therefore, we recommend that you use Db2 compression in your target database during import.

You can define the following types of compression.

Value compression

Value compression stores row data in an optimized format. This is enabled by default for SAP applications, so you do not need to put special consideration on it during database import.

Table compression

Table compression uses a dictionary-based approach to replace repeating patterns with short symbols and includes the following types:

- ▶ Table-wide level
- ▶ Data page level

The table wide level is referred to as classic compression, or often also as static compression. It incorporates a table-wide compression dictionary, which replaces repeating data patterns with shorter symbols. Rebuilding the compression dictionary and recompressing data usually requires a reorganization activity, which is why table-wide compression is also called static compression.

When you perform a database import, you can generate an optimal dictionary at import time by choosing the corresponding R3load option. The advantages and disadvantages between optimal import time and optimal compression dictionary are described in other sections of this publication.

The page level compression is also referred to as adaptive compression and adds to classic compression. It also replaces repeating data patterns with shorter symbols, but replaces them at a page level with a compression dictionary stored inside each data page. The process of creating the page level dictionaries and compressing data inside a page is fully automatic, so other than enabling adaptive compression, no further considerations are required during import of the database. Table compression is enabled on a per-table basis.

Note: You can enable classic compression for a table or you can enable adaptive compression, which also enables classic compression. By default, SAP installations typically use adaptive compression if you decide to compress a table.

Index compression

Index compression is fully automated and is a combination of different techniques. It is enabled by default for compressed tables and requires no further considerations during database import.

For more information, see [IBM Documentation for Db2](#) and search for “Compression” or refer to the following SAP Notes:

- 1126127 – DB6: Deferred table creation and row compression
- 1354186 - DB6: LONG/LOB type mapping and database object check in DDIC
- 2481425 - DB6: How to calculate current LOB size for INLINE LENGTH

6.5.2 R3load compression options overview

The R3load tool offers options for enablement of compression that can affect the estimated runtime and quality of the compression rate.

No compression

By using the COMPRESS_NO_ALL option, the tables are created with the COMPRESS NO attribute. Therefore, table data is not compressed. Also, the corresponding indexes will not be compressed. This is shown in Figure 6-2.

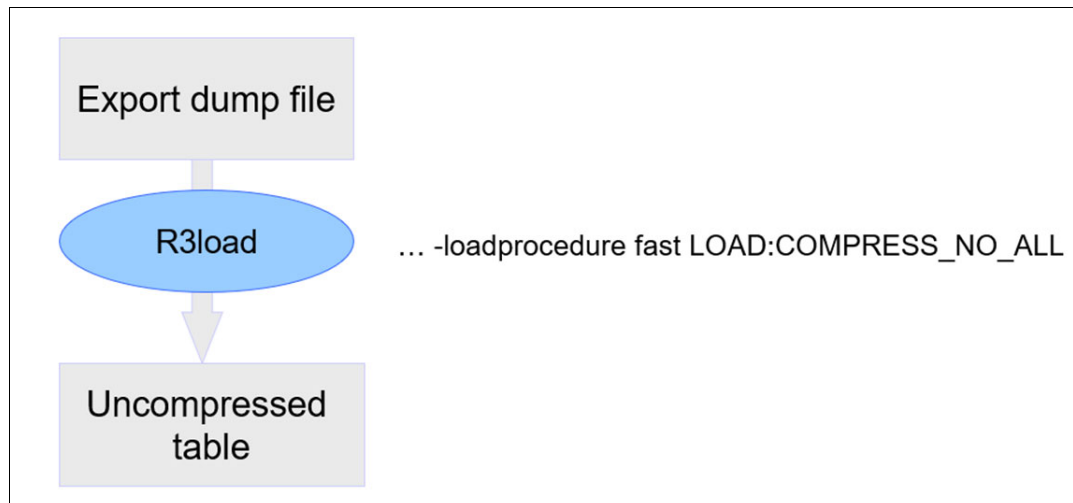


Figure 6-2 R3load without compression

Automatic compression

By using the COMPRESS_ALL option, all tables are created with the COMPRESS attribute. This means that both classic (static) and adaptive compression are used. All indexes of the tables are also compressed. Using the R3load option COMPRESS_ADAPTIVE_ALL is equivalent to COMPRESS_ALL.

The table-wide compression dictionary is created by using the automatic dictionary creation (ADC) feature, which creates the dictionary after a few megabytes of data are loaded. Automatic dictionary creation is typically a good compromise between space savings and import runtime. This is shown in Figure 6-3 on page 98.

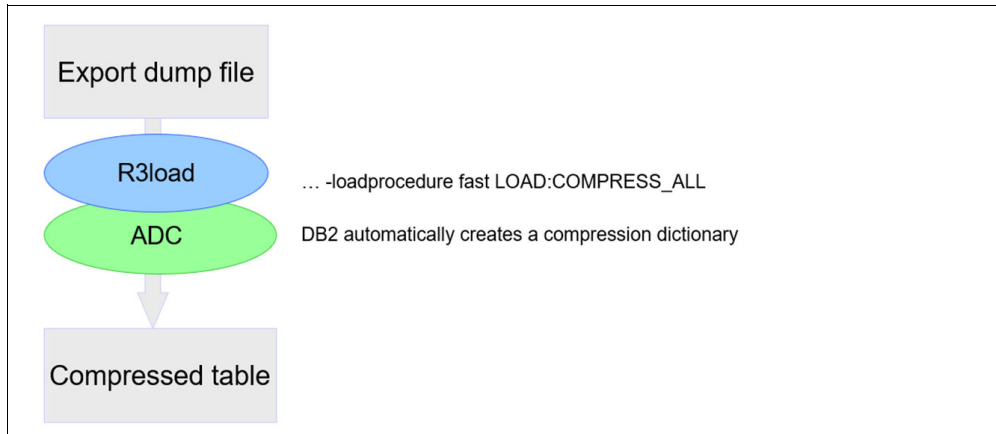


Figure 6-3 R3load with automatic compression dictionary creation

Iterative enablement of compression

By using the FULL_COMPRESS[_ALL] options, tables are compressed with both classic (static) and adaptive compression. See Figure 6-4.

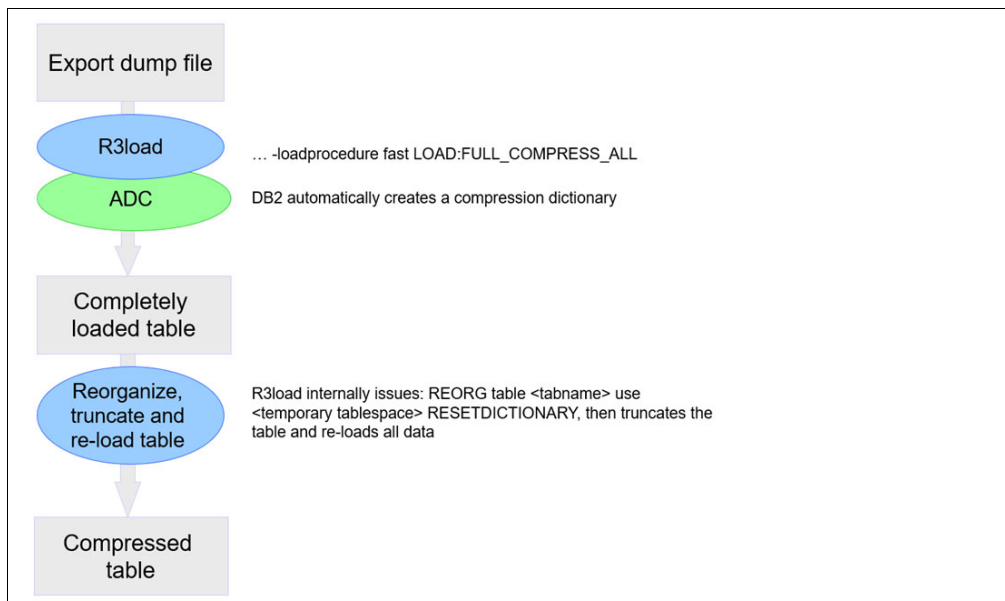


Figure 6-4 R3load with compression dictionary on whole table

The static compression dictionary is built by a table REORG after all data has been loaded:

1. The complete table is loaded in an uncompressed format.
2. The compression dictionary is built.
3. All data is truncated and re-loaded.

Although static compression provides the optimal compression dictionary, the table allocates space for the uncompressed size at first. In addition, reorganization, truncation and re-loading of data increases the import runtime significantly. Therefore, this option is *not* recommended. Use this option for specific scenarios only.

OPT_COMPRESS[_ALL]

The OPT_COMPRESS[_ALL] option is based on a sampling approach. Instead of loading the complete set of data before building the compression dictionary, only a data sample is loaded. By default, the sample size is 1% of the overall data, but this can be adapted by using the DB6LOAD_SAMPLING_FREQUENCY environment variable. The default for this variable is 100, resulting in every 100th record to be used and this is equivalent to 1% of the data. Sample sizes larger than 1% do not usually improve the quality of the compression dictionary.

The second option to influence the sample size is the environment variable DB6LOAD_MAX_SAMPLE_SIZE. The default value is infinity, which means all available data is sampled and loaded. It is possible to specify a number for the number of megabytes sampled. After this amount of data is loaded in the sampling phase, the dictionary is built.

Both options must be set as environment variables and are valid for all R3load processes started in this session or migration monitor instance.

After the data sample is loaded, a reorganization is performed to build the static compression dictionary. The table is then truncated, and the full set of data is loaded. This option results in a near ideal compression dictionary but might increase import runtime. This is shown in Figure 6-5.

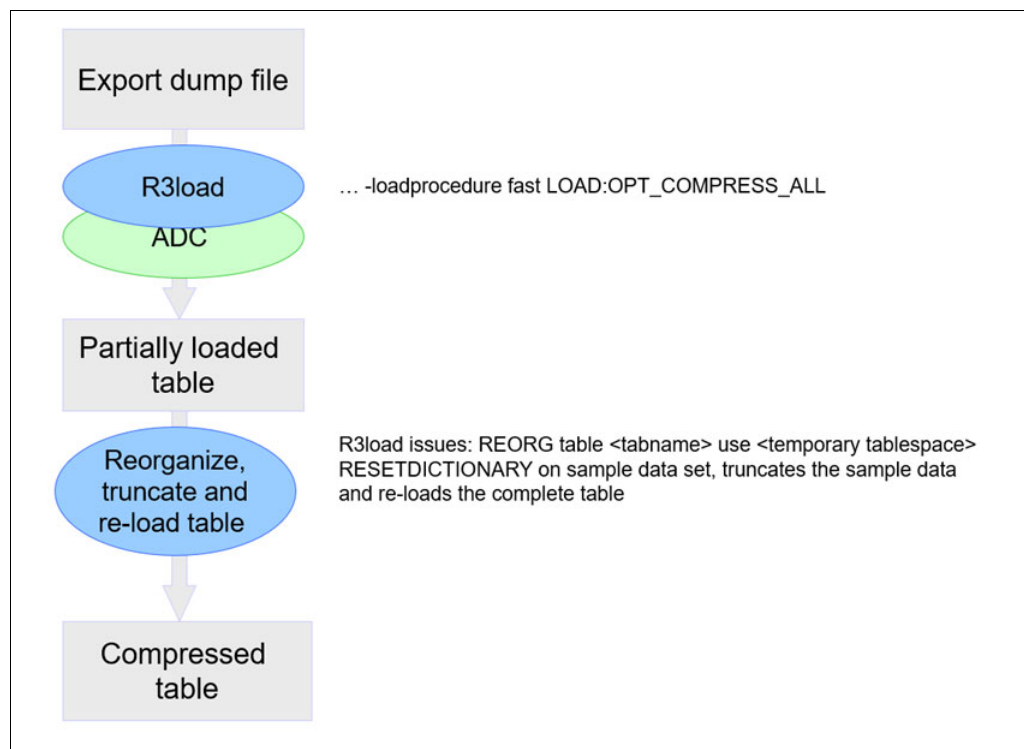


Figure 6-5 R3load with compression dictionary on sample data

The amount of data sampled can have an impact on the reorganization process to build the compression dictionary. However, for a table with one Terabyte of data, the table sample is just 10 GB and the reorganization process should complete in under 5 minutes.

There is a special use case in which limiting the size of the sample that is used can provide some benefit. This is related to the way that the OPT_COMPRESS feature processes data. During the sampling phase, all available R3load dump files are read, data is decompressed and codepage conversion may occur. This is done for all records in the R3load export dump, but only one out of 100 records are written into the database table.

For tables that contain LOB Columns you can use the B6LOAD_MAX_SAMPLE_SIZE parameter to reduce the amount of CPU usage and the number of reads done during the processing of the R3load Dump file. This is particularly true if the specified maximum size is reached after half of the R3load dump files are read, the sampling phase stops, and the remaining 50% of the R3load dump files are not processed during this phase. Example 6-5 shows an import of a table with the default sampling size. With the default setting, the import and reorganization of the sample took 198 seconds, and the final import completed in 1385 seconds.

Example 6-5 R3load with default sampling size

```
A1 EDB6 029 START REORG: 20230125191116
Table VBEP has been reorganized to create compression dictionary
END REORG: 20230125191127
1291583 out of 129158245 rows with maximum row size 696 have been sampled
Finished import for object "VBEP" of type "table" in 198.875 sec, requesting
repeat
Finished import for object "VBEP" of type "table" in 1385.233
```

For comparison, the same table is imported with a max sample size of 4 MB. The results are shown in Example 6-6.

Example 6-6 R3load with sampling size of 4 MB

```
export DB6LOAD_MAX_SAMPLE_SIZE=4
START REORG: 20230126002954
Table VBEP has been reorganized to create compression dictionary
A1 EDB6 017 END REORG: 20230126002955
The data sample is generated with frequency 100 and maximum sample size 4 MB.
6177 out of 617665 rows with maximum row size 696 have been sampled.
Finished import for object "VBEP" of type "table" in 2.457 sec, requesting
repeat
A2 ETSK 005 Finished import for object "VBEP" of type "table" in 1405.233
```

In Example 6-6 the import and reorganization of the sample took only 2.5 seconds and the final import completed in 1405 seconds. The final import times are almost the same, but the difference is in the sampling phase. With the default setting, the sampling phase took 198 seconds, which includes a reorganization time of 9 seconds. With the small sampling size, the sampling phase takes 2.5 seconds, which includes a reorganization time of 1 second. So, the largest part of the reduction is the sample load and not the reorganization process.

This setting can be an optimization when the sampling phase of the OPT_COMPRESS Option significantly contributes to the overall runtime. If you use this option, the compression rate might degrade for two primary reasons:

1. The compression sample is becoming smaller and reaches the size of the default compression dictionary of Db2 automatic dictionary creation (ADC) which is 2 MB.
2. When tables are exported sorted, the sample may include many similar records and will be less efficient.

Special options for compression

When you use the LIST_COMPRESS option, tables listed in the db6_compression_tablist.txt file are compressed with both static and adaptive compression by default. This file is built with one table per line in the file. On each line, you can include a compression option following the table name.

For compression options, the following values are possible:

- YES (default)
- NO
- STATIC
- ADAPTIVE, which is equivalent to YES.

Tables which are not listed in the file are not compressed, unless CREATE TABLE statements contained in the *.SQL files already contain a COMPRESS attribute.

You can use this option to enable specific tables without compression, adaptive compression, or static compression alone. Alternatively, you can specify different compression options for each package in the OderBy.txt file. Because the use of static compression alone is not a recommended or often used option, the LIST_COMPRESS parameter is likely to be used only in certain special cases.

Deferred table creation

SAP systems consist of several thousands of tables, many of them are empty. Every table created in a database consumes a certain amount of space, even if it is empty. For a Db2 database, creating a table in a DMS tablespace allocates the following number of extents:

- 2 extents for the table
- 3 extents for the first index on the table
- 4 extents for LOB objects (if applicable)

Unused tables inside SAP applications lead to unnecessarily allocated disk space. To address this, SAP has introduced the concept of virtual tables. When installing or migrating an SAP system, database virtual tables can be created instead of physical tables. Virtual tables are read-only views that are defined on base tables with no physical representation in the database. When an application attempts to write data into one of these views, the virtual table is defined as a physical table.

The R3load option DEF_CRT enables deferred table creation. It can be combined with other options, such as compression. The following examples describe how to enable deferred table creation.

Example 6-7 shows how to use the deferred table creation option with R3load.

Example 6-7 R3load with Deferred Table Creation Option

```
R3load -i 0005_STXL.cmd -loadprocedure fast  
LOAD:OPT_COMPRESS_ALL:ANY_ORDER:DEF_CRT
```

Initially, R3load creates a view STXL. When there is no data in the export dump file, R3load exits.

When loading data, Db2 returns a SQL Error SQL0150N that can be ignored. Afterward, the view is converted to a table and data is imported. Any compression flag that is used is applied. See Example 6-8.

Example 6-8 R3load with Deferred Table Creation Option – Log File

```
Finished create for object "STXL" of type "table" in 0.047 sec  
Starting import for object "STXL" of type "table"  
[IBM][CLI Driver][DB2/LINUX8664] SQL0150N  
Virtual Table STXL has been successfully converted (ignore SQL0150N)
```

Deferred table creation is beneficial and a best practice. For more information, see *SAP Note "1058437 - DB6: R3load options for compact installation"*.

Like other optimization options, you can use different R3load compression options with the `Orderby.txt` file or dedicated migration monitor instances. As an example, you can use `COMPRESS_ALL` for most tables and `OPT_COMPRESS` for a subset of the tables.

The next sections provide some analysis and recommendations for the use of the import optimizations described.

6.5.3 Compression rates

This book focuses on optimization of heterogeneous system copies. Optimization often means performance optimization to achieve a minimal downtime. Another area of optimization is the best possible compression rate to minimize the database size and to some extent, also improve import performance.

The R3load option `COMPRESS_ALL` provides Db2 automatic dictionary creation (ADC) and provides a good, but not optimal, compression rate.

The R3load option `FULL_COMPRESS[ALL]` provides the best compression rate when the table is imported and then reorganized, and a new compression dictionary is created.

The R3load option `OPT_COMPRESS` provides an optimized compression process and also delivers excellent compression rates that are close to the optimal compression rate.

Figure 6-6 shows a comparison of the compression options on a database. The test database is a little larger than 2 TB without the use of compression.

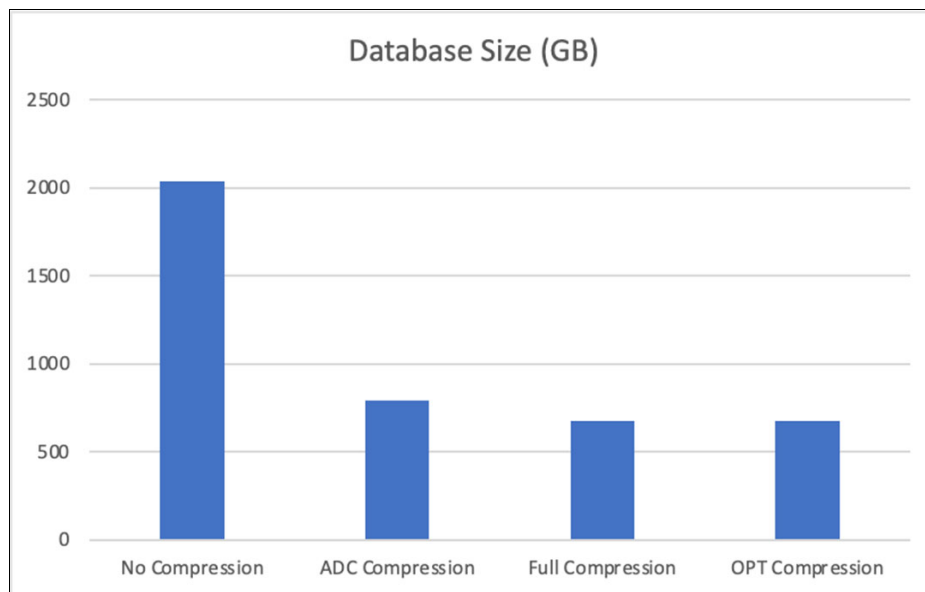


Figure 6-6 Size Comparison of R3load Compression Options

The usage of ADC compression reduces the size to 800 GB and the options `FULL_COMPRESS` and `OPT_COMPRESS` further reduces the size to 675 GB, which is an additional 15% better compression rate. Also, full compression requires more disk space because data is imported first without compression enabled. The allocated disk space cannot be used by the file system until it is deallocated manually. The same is true if you do not enable compression during the migration but enable it after the system is copied.

6.5.4 Performance considerations

Figure 6-7 provides a test case showing the performance implications of the different compression options with the following results:

- ▶ ADC compression is the best option.
- ▶ Full compression requires almost double the amount of import time.
- ▶ OPT compression requires about 40% more import time.
- ▶ The import without compression enabled is slightly slower than ADC compression.

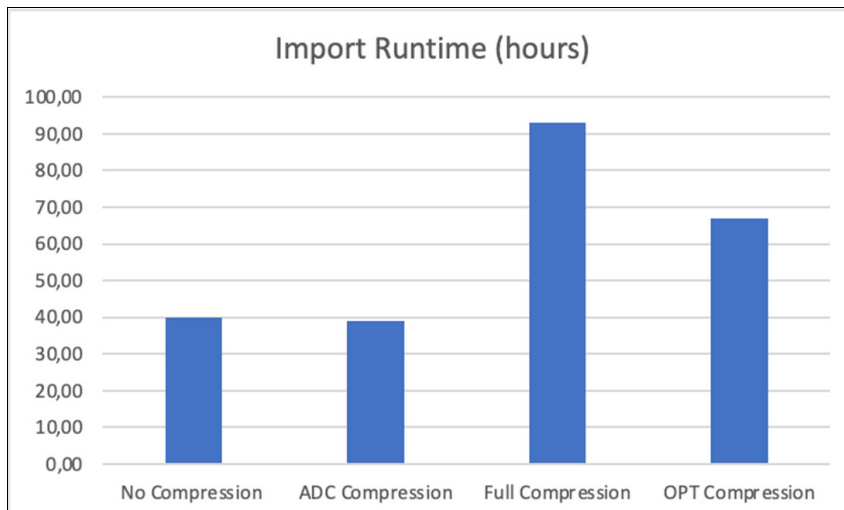


Figure 6-7 Runtime Comparison of R3load Compression Options

Important: The listed results are not absolute numbers that you achieve in every case. The compression rates and the import runtime vary based on the amount and type of large tables and the system resources available.

There are more options available to implement optimal compression, but those require additional manual configuration, and the additional improvement in compression rates might be minimal. Therefore, we do not recommend using these options unless a 100% optimal compression rate is required,

You can reuse the table after a test import if an optimal compression dictionary is built, for example, by a reorganization with the FULL_COMPRESS option or by any other manual sampling. However, the database must be reused and the table must be manually truncated. To accomplish this, you must change the task file. Although this approach is valid, it is complex and error prone. Thus, we do not recommend it and do not provide any additional information about it in this book.

One method of enabling or optimizing Db2 compression can always be used with all of the R3load Options. It is to rebuild a compression dictionary after the heterogeneous system copy is completed. You can do this in the SAP DBA Cockpit or issue the Db2 command REORG TABLE <SCHEMANAME>.<TABLENAME> RESETDICTIONARY. Alternatively, you can use the report DB6CONV in your SAP application to perform a table conversion to optimize the compression dictionary.

To determine whether more compression is savings are possible, you can use the SAP DBA Cockpit. Select **Space** → **Compression Candidates** to analyze potential compression improvements. If the SAP System is not started, you can also use the Db2 ADMIN_GET_TAB_COMPRESS_INFO table function to assess potential improvements.

6.5.5 Tables with large binary objects

Db2 table compression does not compress data in large binary objects (LOB) fields of a table. LOB refers to the BLOB, CLOB, or DBCLOB data types. On a basic level, LOBs are stored on disk level with a reference to this location contained at row level.

Db2 also does not compress data in long varchar fields that have been used prior to LOB fields in a table.

Large tables that are known to contain LOB fields are cluster tables or INDX-like tables, such as STXL, DBTABLOG, COVREF. The SAP data type for such columns is typically RAW or LRAW.

SAP uses a feature of Db2, which is known as *LOB Inlining*. With this feature, for each LOB column in the table, a part of the data is stored within the page and can also be compressed. This piece of the table can be technically compressed, but the compression algorithm does not work efficiently if the data is already compressed by the application.

Consequently, compression rates for tables with a large portion of the data in a LOB column might not be helpful.

Figure 6-8 depicts the compression results for the table STXL. On the primary axis is the compressed size of the table. The table has a total size of 52 GB without compression. With the most efficient way of enabling compression, the size is 49 GB, which is a reduction of only 6%.

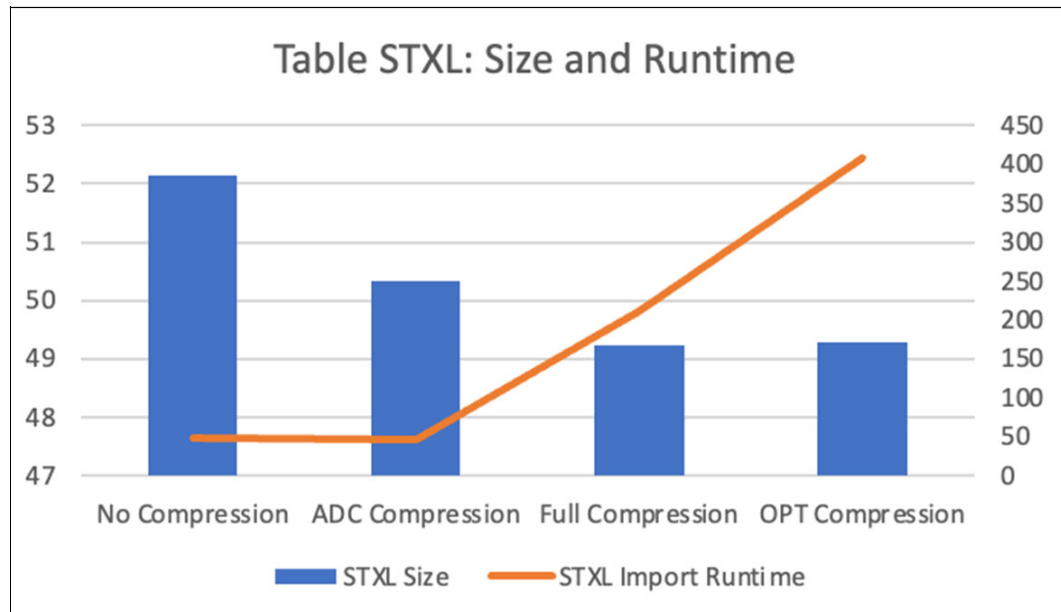


Figure 6-8 Example: Compression results and runtime for different R3load Options.

On the secondary axis, you can see the runtime of the import. Without compression or ADC compression, the runtime is about 50 minutes but dramatically increases with full compression or opt compression. With full compression, the subsequent reorganization contributes to the runtime. With the OPT_COMPRESS method, the R3load dump files are completely read twice, decompressed by R3load and samples are imported. If the dump files reside on slow disks, as they do in our example, this significantly extends the import runtime.

Based on these results, our recommendation for tables with LOB fields is to use ADC compression as default and switch to Full compression or OPT compression only if additional compression may be beneficial.

To do this, run the first test migration with ADC compression and analyze the tables after they are imported.

Note: Because the database can be declustered during the export, perform the analysis on the target system because some tables with LOB Fields, such as RFBLG, might no longer exist on the target system.

Typically, the largest tables determine the overall compression rate. Therefore, you should analyze only the largest tables. You can use the SQL statements shown in Example 6-9 to identify the 25 largest tables.

Example 6-9 SQL Statement to identify tables with column types that are not compressed

```
SELECT SUBSTR(A.TABNAME,1,18) AS TABNAME, SUBSTR(A.COLNAME,1,18) AS COLNAME,
SUBSTR(A.TYPENAME,1,18) AS COLTYPE, B.NPAGES
FROM SYSCAT.COLUMNS A, SYSCAT.TABLES B
WHERE A.TABSCHEMA = B.TABSCHEMA
AND A.TABNAME = B.TABNAME
AND A.TYPENAME IN ('BLOB','CLOB','DBCLOB','LONG VARCHAR')
AND A.TABSCHEMA = '<TABSCHEMA>'
AND B.TYPE = 'T'
ORDER BY B.NPAGES DESC
FETCH FIRST 25 ROWS ONLY
```

Run this statement on the target database because table clusters are declustered and might not contain LOB columns. For example, the table cluster CDCLS contains a RAW field but is converted to the tables CDPOS and PCDPOS that no longer contain LOB columns anymore. Replace `<TABSCHEMA>` with the table schema in your database and change the number of rows to fetch if you want to analyze more tables.

The result is a list of maximum 25 tables for further analysis as shown in Example 6-10.

Example 6-10 Sample output of SQL Query for tables with column types that are not compressed

TABNAME	COLNAME	COLTYPE	NPAGES
STXL	CLUSTD	BLOB	2101600
DBTABLOG	LOGDATA	BLOB	1750537
COVREF	CLUSTD	BLOB	311907
SOFFCONT1	CLUSTD	BLOB	297153
DDNTF	FIELDS	BLOB	10569

In a second step, analyze the tables with the Db2 table function `ADMIN_GET_TAB_COMPRESS_INFO` as shown in Example 6-11.

Example 6-11 SQL Statement to identify compression savings

```
SELECT SUBSTR(TABSCHEMA, 1, 10) AS TABSCHEMA, SUBSTR(TABNAME, 1, 10) AS TABNAME,
PCTPAGESSAVED_CURRENT,
PCTPAGESSAVED_ADAPTIVE
FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO(<TABSCHEMA>, '<TABNAME>'))
```

As a result of the query, you get the estimated compression savings for the table if you perform a reorganization. Replace <TABSCHEMA> with the SAP Schema and <TABNAME> with the table to be analyzed. The reported amount that is saved is also close to the compression results you can get with the R3load Options FULL_COMPRESS or OPT_COMPRESS.

The query results are shown in Example 6-12. The tables STXL and DBTABLOG might be candidates for optimized compression.

Example 6-12 Sample output of SQL Query to assess further compression savings

TABSCHEMA	TABNAME	PCTPAGESSAVED_CURRENT	PCTPAGESSAVED_ADAPTIVE
SAPSR3	DBTABLOG	52	84
SAPSR3	STXL	4	7

The tables have been imported using ADC Compression and in this example, the table DBTABLOG is a candidate for further optimization as significant savings are possible. However, the table STXL does not benefit from enhanced compression.

To optimize the compression process without increasing the runtime, you can use OPT_COMPRESS as a global option and specify the option COMPRESS, which equals to ADC, in the OrderBy.txt file for the table STXL and for all others that do not benefit from compression.

6.5.6 Conclusion

Table 6-3 summarizes the advantages and disadvantages of the different options for Db2 compression.

Table 6-3 Comparison of R3load Compression Options

R3load Option	Compression effect	Import runtime	Tables grow to max size (before compression)
COMPRESS_NO_ALL	none	optimal	yes
COMPRESS_ALL	good	optimal	no
FULL_COMPRESS	optimal	high	yes
OPT_COMPRESS	optimal	moderate to high	no

Our recommendation is to use ADC compression, the COMPRESS_ALL option, as a default. If you want to achieve an optimal compression rate, use OPT_COMPRESS but pay attention to tables with LOB Columns because import runtime might increase.

6.6 Import optimization

The data import phase is a long-running task. Therefore, it is important to optimize the import using SAP tools and Db2 configuration settings. The following section provides information

about optimization techniques and some insight and background information about configuration settings.

Although all information provided here is based on real-life migrations and intensive tests, there is no specific set of parameters and values available that apply to all migrations. The parameters must be adopted for each environment.

Db2 provides different methods of populating tables with a bulk of data. The options are Db2 LOAD Command, the db2load API, the INGEST Command, the IMPORT Command, and SQL INSERT with different features for optimization. SAP uses only the db2load API and INSERT in R3load. Therefore, the word *import* is used as a generic term for populating tables, irrespective of the method used. The method is either the db2load API or INSERT.

6.6.1 Db2 LOAD versus INSERT

You can populate tables in Db2 by using the following methods:

- ▶ INGEST
- ▶ IMPORT
- ▶ INSERT
- ▶ LOAD

SAP uses only INSERT or the db2load API. The INSERT procedure is sufficient for installations and migrations of smaller databases and is sometimes the better choice. If a large amount of data must be moved from one system to another system, the db2load API is faster because it writes formatted pages directly into the database.

The processing with Db2 LOAD is optimized on throughput. The overall load process is divided in several distinct phases: loading, building indexes and statistics, delete phase and the index copy phase. R3load uses the db2load API with a subset of the available Db2 LOAD features. Therefore, only two phases are normally shown in the Db2 diagnostic log file when using R3load:

1. Load. Phase during which the data is written to the table
2. Build. Phase during which existing indexes are created

Note: The data imported with the db2load API is not logged. Therefore, a subsequent roll forward marks all tables populated by Db2 LOAD as not accessible. Therefore, a Db2 backup after a successful import is mandatory!

Data processing by using Db2 LOAD versus INSERT

The two different methods of populating a table have different characteristics and it is valuable to understand those differences to decide which method or optimization to use.

Locking

The locking behavior of Db2 LOAD and INSERT is different. Db2 LOAD locks the table exclusively, and insert uses row level locks unless lock escalations occur. If multiple R3load processes use the db2load API to import data into the same table, for example if there is a split table, only one of them can import data at a time. If multiple R3load processes use INSERT to import data, they can import the data concurrently.

Codepage conversion

When the data contains UTF16 encoded character data (UNICODE), the data must be converted from UTF16 to UTF8. When you use INSERT, this conversion is performed on the

Db2 client side. When you use Db2 LOAD this conversion is performed by the db2load API on the database server.

► Logging

When you use Db2 LOAD, the amount of logging is reduced. For each new extent that is created in the table, only one log entry is written. When you use INSERT, each loaded record is logged unless logging is disabled.

► Triggers and constraints

The Db2 LOAD utility does not fire triggers and does not perform referential or table constraints checking, other than checking the uniqueness of indexes.

Performance comparison – Db2 LOAD versus INSERT

Db2 LOAD is typically much faster than INSERT. The increase of throughput with Db2 LOAD depends on the underlying infrastructure and to some extent also on the table structure. If your system is I/O constrained, Db2 LOAD can show less improvement as the disks might not be able to write the data as fast as requested.

In a test with table SWWLOGHIST (60 million records, 25 GB data), we compared an import using INSERT with an import using the db2load API. By using INSERT, it took approximately 12 hours and 30 minutes to insert all data into the table. By using LOAD, the same table was loaded within 32 minutes or about 20 times faster.

Figure 6-9 includes different test results. Db2 LOAD typically shows significant performance improvements over INSERT for most of the tables like GLPCA and SWWLOGHIST.

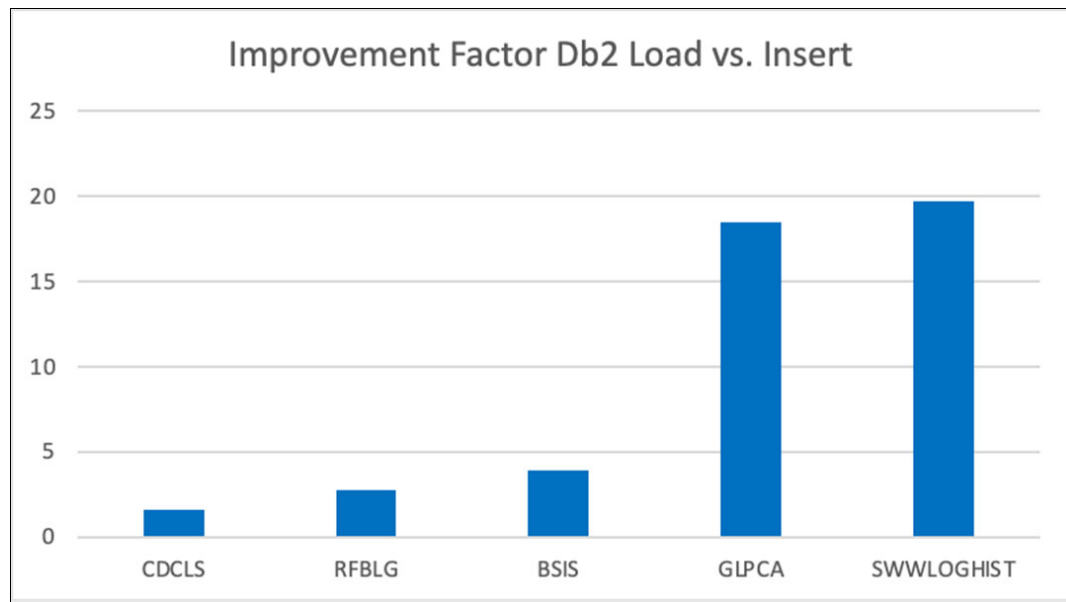


Figure 6-9 Runtime Comparison – Db2 LOAD versus INSERT (Improvement Factor)

However, the improvement factor is less on tables CDCLS, RFBLG, or BSIS. Tables CDCLS, RFBLG, or BSIS might be a candidate for INSERT with the table split option that allows multiple R3load processes to populate the table in parallel. However, the db2load API is usually twice as fast compared to a single INSERT stream, even in the worst case scenario. Therefore, the db2load API is the default for R3load in SWPM.

The advantage of the db2load API is significant for tables with a few columns like VBOX, CDHDR, ACCTCR, or SWWLOGHIST.

There are two classes of tables with less improvement when using the db2load API:

1. For tables with many columns, for example 90 and more, the performance advantage can be less. Examples for such tables are BSIS, COEP and ACCTIT.
2. Another class of tables that are typically only a factor of 2–5 times faster compared to insert are tables with LOB columns where most of the data is not Inlined. Typical examples for this are cluster tables, INDX-like tables, or tables with LRAW 32000 fields. Tables in this category are CE1IDEA, SOFFCONT1, and STXL.

Small tables with less than 200 KB of data do not benefit from Db2 LOAD. In this case, the CPU usage during initialization of the utility is high compared to the performance improvement. Because of this factor, you will see tables imported by R3load with INSERT, even if you specify the use of the db2load API. R3load automatically switches to INSERT for small tables.

Table 6-4 can be used as an example to assess the differences in throughput for Db2 Load. These are just examples and should not be considered as a reference for your environment. The actual throughput depends on the IT infrastructure, the schedule of R3load Jobs and table content.

Table 6-4 Sample Db2 LOAD Throughput

Table	Rows/Sec	MBps
SOFFCONT1	1280.00	39.20
VRSMODISRC	7727.00	33.31
VBAP	11842.00	53.22
ACCTIT	12355.92	33.82
BSEG	13798.00	54.64
MSEG	14058.00	56.63
CE1IDEA	16499.00	36.44
STXL	20781.00	160.37
DBTABLOG	20821.00	334.08
BKPF	38705.00	63.05
COEP	47907.00	70.08
VBEP	71271.00	47.31
BSIS	84262.94	116.84
EDIDC	85649.00	89.52
FAGLFLEXA	97273.00	55.29
ACCTCR	155407.87	17.19
VBPA	194784.00	67.62
CDHDR	434780.14	168.34
VBOX	439173.00	71.20

R3load options for Db2 LOAD and INSERT

There are different R3load options available to use the different approaches to import data. The default option used by SWPM is to enable Db2 LOAD and we recommend using this default.

The following option can be specified to use INSERT:

loadprocedure fast INSERT with optimizations

The following option can be specified to use Db2 LOAD:

loadprocedure fast Load Uses Db2 LOAD utility for all viable procedures

Note: If you want to use Db2 LOAD instead of INSERT, you can use the R3load Parameter `LOAD_FORCED` to use LOAD in any case. A viable scenario for this option is described in section 7.3, “Table splitting” on page 139.

For more information about the various R3load options, see *SAP Note 454173 - DB6: Accelerated R3load migration through CLI LOAD*.

6.7 Using and optimizing Db2 LOAD with R3load

There are several options available to optimize the R3load processing. Some options can be enabled by R3load parameters. Other options require a change in the Db2 database or Db2 database manager configuration. Another set requires the use of environment variables.

Tip: As a best practice, do not set the environment variable temporarily in the user environment. Create a copy of the `import_monitor` script and add the environment variable there. Another option is to use a dedicated script that sets the environment variable and calls the `import_monitor` script.

6.7.1 Configuring the db2load API

There are several options for R3load to control the behavior of the `db2load` API. In addition to the options mentioned previously, the following environment variable settings are supported:

- ▶ `DB6LOAD_CPU_PARALLELISM=<n>`
This variable controls the Db2 LOAD `CPU_PARALLELISM` parameter.
- ▶ `DB6LOAD_DATA_BUFFER_SIZE=<n>`
This variable controls the Db2 LOAD `BUFFER_SIZE` parameter.
- ▶ `DB6LOAD_DISK_PARALLELISM=<n>`
This variable controls the Db2 LOAD `DISK_PARALLELISM` parameter.

- ▶ `DB6LOAD_INDEXING_MODE=<n>`
This variable controls the Db2 LOAD INDEXING_MODE parameter, where 0 equals AUTOSELECT (Default), 1 equals REBUILD, 2 equals INCREMENTAL and 3 equals DEFERRED.
- ▶ `DB6LOAD_FORCE_LOAD`
This variable forces Db2 to use Db2 LOAD even for small or split tables if it is set to any value, such as 1). You can also specify the force option directly in the R3load arguments as `LOAD_FORCED`.

6.7.2 DB6LOAD_CPU_PARALLELISM

Use this parameter to exploit intra-partition parallelism. The parameter specifies the maximum number of processes or threads used by the db2load API to parse, convert, and format data records. The maximum value allowed is **30**. If this parameter is not specified, the db2load API selects a default value that is based on the number of CPUs on the system, which generally is a good choice.

The LOAD utility is designed to load a small number of tables in parallel and to use a larger number of CPUs. For an SAP heterogeneous system copy, typically 50 or more R3load processes are running in parallel. In addition, experiences and tests have shown that with a CPU parallelism of 4 then 80%–90% of the maximum throughput is achieved.

Therefore, R3load uses a CPU parallelism of 4 as default. This allows for acceptable throughput and ensures that the CPU resources are optimally used.

The throughput decreases significantly if the CPU parallelism is set to **1**, which is not recommended. The effective CPU parallelism can decrease automatically when there is not enough memory available in the Db2 Utility Heap.

This means, even if you set `DB6LOAD_CPU_PARALLELISM` to a dedicated value or use the R3load default, Db2 might decrease this number if not enough memory is available. This is discussed in more detail with recommendations in section 6.7.3, “Db2 utility heap and `DB6LOAD_DATA_BUFFER_SIZE`” on page 112.

Specifying larger CPU parallelism

Some conditions might benefit with a larger CPU parallelism setting.

Larger CPU resources on the target

Typically, there is no reason to specify a larger CPU parallelism. An exception might be if the target system temporarily has a large amount of CPU resources available and if the number of R3load processes are limited. For example, if your target machine has 416 virtual CPUs and you are running with 50 R3load processes in parallel, it might be beneficial to increase the CPU parallelism.

Tables with different parallelism settings

Another use case might be to specify a larger CPU parallelism for the table or a small number of tables that determine the overall import process. This would mean that most of the tables are imported with the default CPU parallelism of 4, and a few tables are imported with a higher degree of parallelism.

To do so, two separate instances of the Migration Monitor with a mutually exclusive set of tables are required. If you want to use this feature, become familiar with the Migration Monitor and how to use it outside the SWPM.

This setup means let the migration monitor instance that is started by SWPM run with the default CPU parallelism of 4 and start a second Instance with a setting greater than 4. To do so, set the environment variable DB6LOAD_CPU_PARALLELISM only in the terminal session for this second instance.

6.7.3 Db2 utility heap and DB6LOAD_DATA_BUFFER_SIZE

The database parameter utility heap size (UTIL_HEAP_SZ) specifies the maximum amount of memory that can be used simultaneously by the Db2 utilities like BACKUP, RESTORE or LOAD. If the parameter is set too low, you might not be able to concurrently run utilities.

Based on the number of processes defined for loading data, the page size, and extent size of the tablespace the table resides in, the memory is calculated for a LOAD operation. This memory is allocated from the utility heap. If there is not enough memory available, the number of CPUs used for LOAD are reduced.

The Db2 Utility Heap can be configured with a fixed value between **16** and **2147483647** 4K pages, which equals a maximum of 8 TB. It can also be set to AUTOMATIC and this means, the memory is allocated within the boundaries specified by the Db2 configuration parameter INSTANCE_MEMORY.

Because a sufficient buffer size for the Db2 LOAD utility is essential for optimal performance, it is a best practice to specify a lower boundary for the Db2 utility together with the automatic setting. A good starting point is 500,000 pages or more by using the following command:

```
UPDATE DB CFG FOR <DBSID> USING UTIL_HEAP_SZ 500000 AUTOMATIC
```

The DB6LOAD_DATA_BUFFER_SIZE parameter specifies the total amount of memory that is allocated to a single Db2 LOAD utility used by R3load as a buffer. If not set, Db2 allocates the memory automatically with a sufficient size. Generally, you do not set DB6LOAD_DATA_BUFFER_SIZE to any specific value.

If the machine is memory constrained to 64 GB or less and you need many parallel R3load processes, consider setting the DB6LOAD_DATA_BUFFER_SIZE environment to a fixed low value and adapt the Db2 Utility Heap. For example, if you want to run 100 R3load processes that use the db2load API in parallel you can set the DB6LOAD_DATA_BUFFER_SIZE=10000 and UTIL_HEAP_SZ=1000000 (10,000 x 100) to accommodate 100 parallel R3load processes.

Figure 6-10 on page 113 shows the effect of a slowdown due to shortage of utility heap and the resultant reduced performance. The table CDCLS was exported in chunks and imported with R3load using the db2load API.

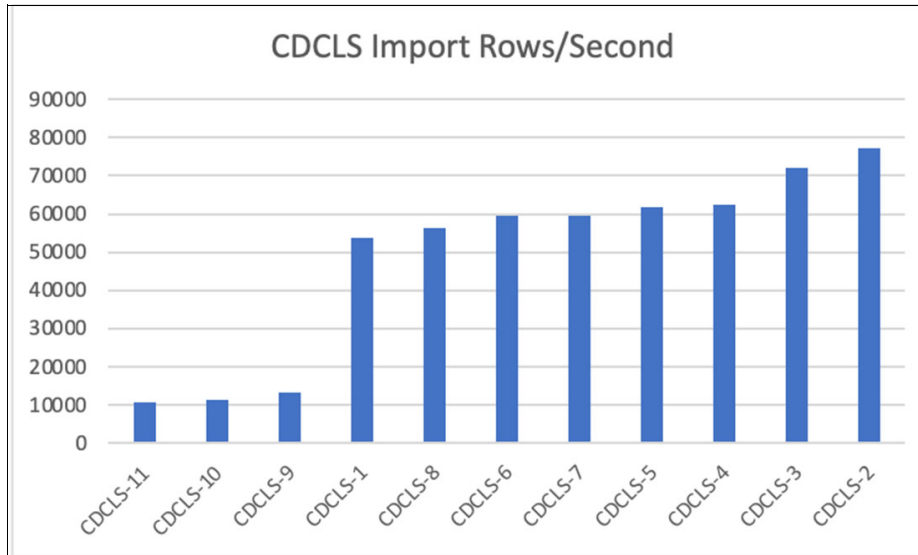


Figure 6-10 Throughput of CDCLS splits with insufficient Utility Heap

The throughput of the table varies significantly. Although the imports of the parts 1–8 achieved over 50,000 rows per second, parts 9–11 were only running with about 10,000 rows per second. During the import, too many R3loads were started in parallel to parts 9–11 and Db2 LOAD did not get enough memory (less than 4,000 4K pages) and therefore reduced the CPU parallelism.

How to identify a slowdown? The Db2 diagnostic log file records the parallelism of each Db2 LOAD operation and so you can check whether the LOAD utility has decreased the CPU Parallelism. This is shown in Example 6-13.

Example 6-13 Messages in Db2 diagnostic log file created by Db2 LOAD

```

2023-01-23-15.23.45.962927+060 I41900E554          LEVEL: Warning
PID      : 1594211          TID : 140351612380928 PROC : db2sysc 0
INSTANCE: db2tar          NODE : 000          DB : TAR
APPHDL  : 0-44          APPID: *LOCAL.db2tar.230123135508
UOWID   : 59          ACTID: 1
AUTHID  : DB2TAR          HOSTNAME: MIGTARGET
EDUID   : 215          EDUNAME: db2agent (TAR) 0
FUNCTION: DB2 UDB, database utilities, sqluInitLoadParallelism, probe:1097
MESSAGE : Load parallelism modified from 4 to 1

```

Important: Ensure that Db2 is not throttling CPU parallelism to a value of one. This can create a significant performance degradation.

The Db2 LOAD Utility also stores the Utility information in the history file. This file also includes information about the memory used for each load operation. In an SAP environment, the Db2 registry variable DB2_HISTORY_FILTER=L is temporarily set by SWPM to reduce potential contention on the history file.

For a detailed analysis, you can temporarily unset this registry variable and the buffers used for each Db2 LOAD process is reported in the Db2 history file.

6.7.4 DISK_PARALLELISM

The DISK_PARALLELISM parameter specifies the number of processes or threads to be used by the LOAD utility to write data records to disk. Use this parameter to improve load performance. The maximum number of threads is the higher value of either four times the CPU_PARALLELISM value used by the LOAD utility or 50.

By default, DISK_PARALLELISM equals the sum of the tablespace containers on all tablespaces that contain objects for the table being loaded except where this value exceeds the maximum number allowed. Typically, there is no need to change this value manually.

6.7.5 Indexing mode

Specifies whether the LOAD utility rebuilds indexes or extends them incrementally. Valid values are:

- ▶ AUTOSELECT (default)

The Db2 LOAD utility automatically selects either REBUILD or INCREMENTAL mode.

- ▶ REBUILD

All indexes are rebuilt. The utility must have sufficient resources to sort all index key parts for both old and appended table data.

- ▶ INCREMENTAL

Indexes are extended with new data.

- ▶ DEFERRED

The LOAD utility does not attempt to create an index if this mode is specified. Indexes are marked as needing a refresh. The first not LOAD related access to DEFERRED indexes forces a rebuild, or indexes might be rebuilt when the database is restarted. However, this option is ignored in many cases as almost all SAP tables have unique indexes. The indexes are defined as part of the database table configuration based on SAP recommendations prior to loading the database.

Changing any of the indexing mode values is usually not recommended. Use the default unless the value must be changed for specific optimizations. One of these specific optimizations is the usage of db2load API when sequentially importing split tables. In this case, setting the indexing mode to INCREMENTAL can help to improve the overall performance of the import, when indexes are created before the load.

6.7.6 Preserving order of the data

The db2load API leverages SMP Parallelism and uses multiple threads to populate a table. As a default, the db2load API preserves the order of that data that is loaded. Therefore, the different threads are synchronized and must wait for each other.

This functionality does not actively sort the data. It verifies the order of the data. So, if an unsorted export is loaded, it remains unsorted. Therefore, it is beneficial to disable preserving the data for all packages that are exported unsorted.

The R3load Option ANY_ORDER enables the db2load API to ignore the order of the data handed to it to be loaded.

Even for tables that are exported with sorting, this parameter can help to improve load performance. The sorting is typically required for the export R3load processing, for example

declustering or codepage conversion, but not for the import side. Therefore, the ANY_ORDER option can be used on the import side for sorted exports as well. This is true also if R3load requires the data to be sorted, such as due to codepage conversion on the target or declustering on the target, because the R3load processing is done before the db2load API processing.

Do not use the ANY_ORDER option if you require strict sorting of data on the target system. For example if you achieve 100% optimal compression rate with minimal additional savings.

6.7.7 Optimal number of R3Load processes

One important tuning option is to select the optimal number of import processes and the optimal number of export processes. The goal is to maximize the usage of all available resources such as CPU, disks, memory, and network. Although the optimization of the memory usage is primarily related to the Db2 configuration, the usage of CPU and disks is influenced primarily by the number of R3load processes that are running in parallel.

To better explain this, an example is provided where a larger number of R3load processes results in decreased throughput.

The Import Time Graphs shows that the migration with 2 parallel load process completes after two hours as shown in Figure 6-11.

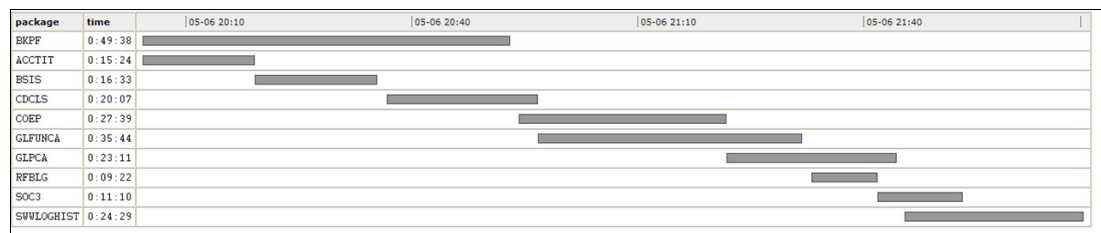


Figure 6-11 Import times with 2 parallel load processes

The same import with 8 parallel processes took 3.5 hours as shown in Figure 6-12.

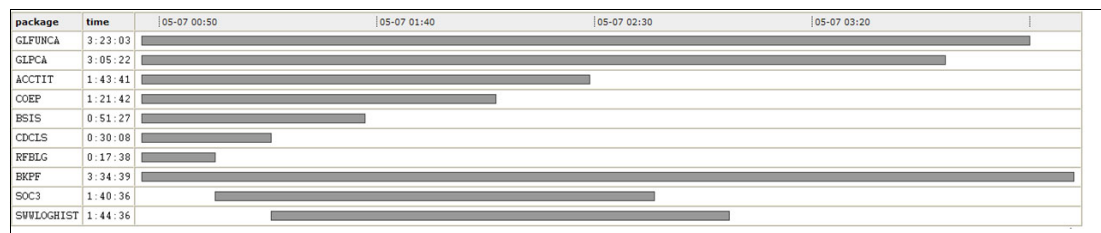


Figure 6-12 Import times with 8 parallel load processes

Looking at the timings, the BKPF package went from 45 minutes to 3.5 hours. Because of this the overall runtime of the import expands. This is caused by using too many parallel processes. The reason for this slowdown is typically overloading resources. In the example shown in Figure 6-12, it is the storage subsystem.

In other cases, it might be the CPU usage. Figure 6-13 on page 116 shows the CPU usage during an import of a single table with R3load and the db2load API with a CPU Parallelism of 8 that is not I/O bound.

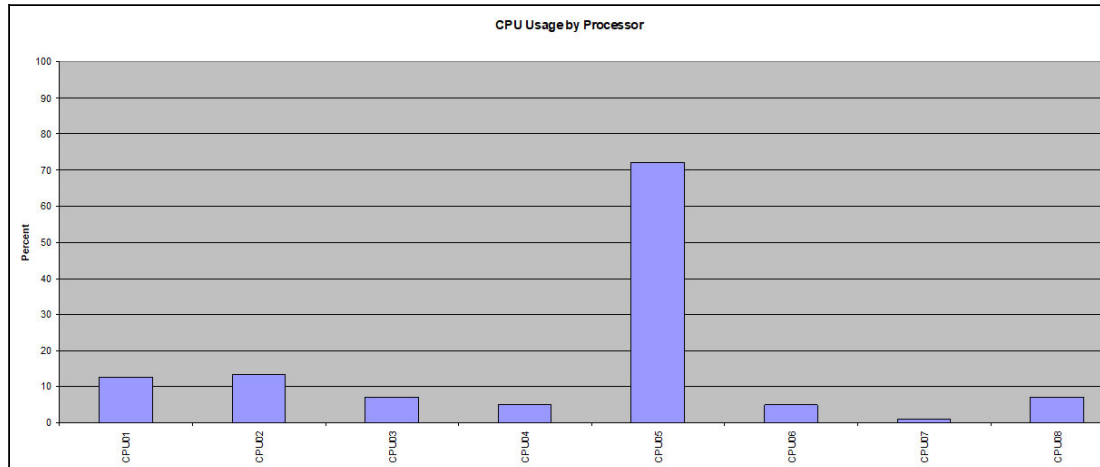


Figure 6-13 CPU Distribution with Db2 LOAD

One CPU has a much higher workload compared to the others. A typical scenario is that one CPU is used at 100% while others are using 10%–20% for a single load process.

This data demonstrates that the number of Importing processes should not be higher than the number of available CPUs. In fact, a good recommendation is to start with only 75%–80% of the CPU count for the maximum number of parallel processes.

For example, if the import server has 128 CPUs, you should not use more than 90–100 parallel R3load processes.

This can be used as a starting configuration, but monitor the resource usage during the migration process, and ensure no resource is overloaded nor resources are idle.

6.7.8 Cleaning up a failed Db2 LOAD

If an R3load that is using the db2load API fails, you might have to terminate a LOAD if the table stays in a LOAD pending state. To terminate a failed LOAD, issue the commands shown in Example 6-14.

Example 6-14 Commands to terminate a failed LOAD

```
db2 connect to <db_name> user <connect user>
db2 load from <empty file> of ixf terminate into <schema>.<tablename>
```

<Empty file> corresponds to any file on the filesystem that has a size of 0 Bytes.

If for any reason the LOAD terminated abnormally, the tablespaces in rare cases remain in quiesce exclusive state. The command `db2 list tablespaces` shows a state of 0x0004. To remove this status, use the commands shown in Example 6-15.

Example 6-15 Db2 command to enable tablespace access

```
db2 connect to <db_name> user <connect user>
db2 quiesce tablespaces for table <failed table> exclusive
db2 quiesce tablespaces for table <failed table> reset
```

This procedure will unquiesce the tablespaces and after this, the table can be cleaned up from the LOAD pending state.

6.8 Order and configuration of index creation

In addition to loading the data into the database, it is important to optimize the creation of indexes. Depending on their structure and size, index creation can consume a large portion of the overall migration time. The goal must be to minimize the time for index creation. Things to consider in this area are the order of creating indexes and memory usage.

6.8.1 Order of index creation

To achieve the best performance for the index creation, try to avoid concurrent index creation and avoid spilling data to disk. This can be influenced by controlling the order of creating indexes, mode of index creation and choice of appropriate Db2 parameters.

The R3load tool allows you to specify whether the indexes should be created before or after the data load. If you create the indexes before the load, they are maintained during the build phase of Db2 LOAD.

Example 6-16 shows a `db2diag.log` with entries for the build phase of the Db2 LOAD.

Example 6-16 Messages in Db2 diagnostic log file created by Db2 LOAD

```
2022-07-20-03.46.17.195667 Instance:db2fcp Node:000
PID:5721(db2lrid) Appid:*LOCAL.db2fcp.010719161634
database_utilities sqlulPrintPhaseMsg Probe:0 Database:FCP
Starting BUILD phase at 07-20-2022 03:46:17.193097.
2022-07-20-03.56.41.297931 Instance:db2fcp Node:000
PID:5721(db2lrid) Appid:*LOCAL.db2fcp.010719161634
database_utilities sqlulPrintPhaseMsg Probe:0 Database:FCP
Completed BUILD phase at 07-20-2022 03:56:41.288616.
```

To decide on the order of index creation, consider the following issues:

- ▶ The performance of the index creation depends to a large extent on the scheduling. If many tables create indexes in parallel, the I/O Subsystem might slow the process. It might be more beneficial to pause the index creation for some tables and schedule the index creation during a period with less I/O contention.
- ▶ If the index is created before the load and if the export dump contains duplicate records, then both the load and the index creation must be repeated. If the index is created after the load and if duplicate records exist, it might be possible to clean up the records and redo the index create only.
- ▶ For a cleanup of a split table load, the data is deleted in multiple chunks using appropriate SQL Statements. If no index exists, the result are multiple slow table scans.
- ▶ The process of maintaining the index is different for tables that are imported with the db2load API or by using INSERT. The db2load API maintains the indexes during a dedicated phase, and the insert maintains the indexes for each record inserted.

The default for indexes creation is `AFTER_LOAD`. Editing the first two lines in the `DDLDB6.TPL` file determines the order of index creation for R3load as shown in Example 6-17.

Example 6-17 DDLDB6.TPL file is used to define the order of index creation

```
prikey: AFTER_LOAD ORDER_BY_PKEY
seckey: AFTER_LOAD
```

When tables are imported using INSERT instead of db2load API, creating the Indexes is faster in all cases when the default of AFTER_LOAD is used. The impact can be significant. See Figure 6-14.

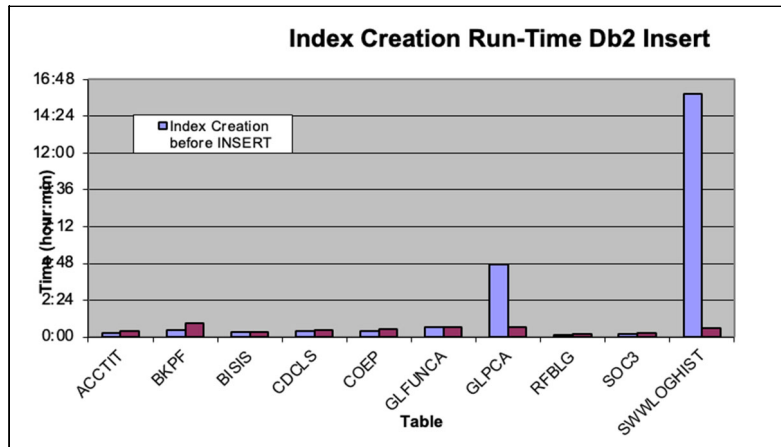


Figure 6-14 Performance impact of index creation before and after with INSERT

The same is true when tables are loaded with the db2load API. In almost all cases, it is faster to create the index after the tables are loaded. Figure 6-15 shows our test results.

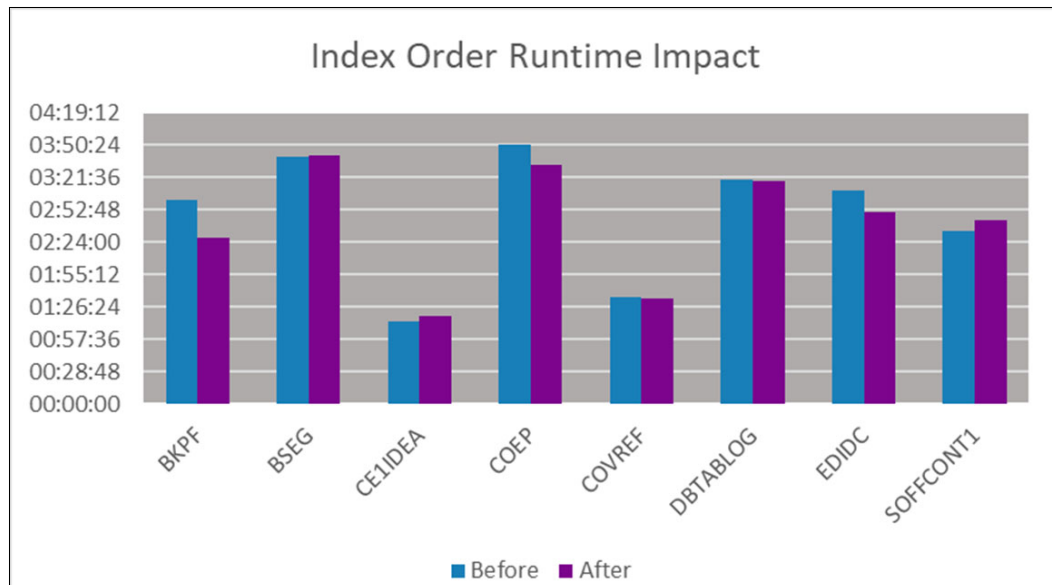


Figure 6-15 Performance impact of index creation before and after with Db2 LOAD

Note: This is true for Db2 10.5 and higher and has changed from previous versions due to optimizations in Db2.

Sometimes, it might be useful to change the order of index creation for the primary index. A test with the tables EDIDC and COEP, for example, had a noticeable improvement of about 10% when the primary key index is built before the table is loaded. This test is not conclusive enough to recommend this combination, but it is worth trying this configuration in case you must optimize the process. You can implement this optimization by creating a special template file, such as DDLDB6_INDX.TPL and use this for specific tables in the same or a dedicated migration monitor instance.

Another scenario where a deviation from the default might make sense is the usage of the Db2 registry variable DB2_SMP_INDEX_CREATE that is explained in section 6.10, “SMP parallelism for index creation” on page 121.

Note: The index creation typically generates the highest I/O load during the migration process. Therefore, a good optimization strategy is to distribute the index creation uniformly over the whole migration process to be able to use all available resources, for example, CPU, disk, and memory.

6.8.2 SORTHEAP

Setting SORTHEAP correctly together with SHEAPTHRES_SHR is one of the most powerful optimization options for the import and for the index create phase. If set correctly, it minimizes I/O, and the speed of the index build can improve to a maximum of 50%.

The SORTHEAP parameter defines the maximum number of private or shared memory pages to be used for a single sort. If you define this parameter as large as possible for the index creation, you might avoid sort overflows and spilling data to disk.

This parameter can be set significantly higher compared to a typical value during production. Although the recommendation for productive operation of ERP systems is at least 2048, SORTHEAP should be set to a significantly larger number during migrations.

Although it is usually not possible to avoid spilled sorts during a migration of large tables, the goal is to avoid as many sort overflows as possible. So, if enough memory is available, set the SORTHEAP parameter to automatic with a starting value of 1,000,000 pages. To optimize this value, monitor the database for sort overflows and adjust the parameter together with the SHEAPTHRES_SHR configuration parameter. Consider assigning available memory to the sort areas instead of buffer pools during target system import.

6.8.3 SHEAPTHRES_SHR

SHEAPTHRES_SHR represents a soft limit for the total amount of database shared memory that can be used by sort memory consumers at any time whereas SHEAPTHRES defines the limit on instance level. SAP recommends that SHEAPTHRES be set to 0. This means that all shared sorts are performed within the database memory.

Ideally, set SHEAPTHRES_SHR to automatic with a reasonable multiple value of SORTHEAP as initial value.

A good starting configuration is to set the value of SHEAPTHRES_SHR to the number of parallel R3load processes multiplied by the minimum SORTHEAP value. However, the optimal value might be higher because indexes are created in parallel, and so multiple parallel sorts for a single index create might run in parallel.

To monitor the correct setting of the SORTHEAP and SHEAPTHRES_SHR parameter, you must monitor the sort overflow in the database.

As a starting point, you can use the MON_GET_DATABASE table function or DBSUMMARY procedure that is described in Chapter 10, “Tips for monitoring” on page 175. In both cases, you must analyze if sort operations are spilling to disk. The reported values for the following metrics should decrease or are zero after optimization of the sort memory:

sort_overflows	The total number of sorts that ran out of sort heap and might have required disk space for temporary storage.
-----------------------	---

- post_threshold_sorts** The number of sorts that have requested heaps after the sort heap threshold has been exceeded.
- post_shrthreshold_sorts** The total number of sorts that were throttled by the sort-memory throttling algorithm. A throttled sort is a sort that was granted less memory than requested by the sort-memory manager.

Note: Most likely, you cannot decrease these sort metrics to zero as the build of indexes for large tables might exceed the available memory on the system. However, remove as many spilled sorts for other large and medium-sized tables as possible to optimize the resource consumption on the storage subsystem.

6.9 Db2 buffer pools

The db2load API does not use the buffer pool for asynchronous writes. However, buffer pools are used during index creation and inserts. When you configure buffer pools, consider that buffer pools and other parameters, such as utility heap and sort heap, are allocated from the same available memory.

The only workload on the database during the import phase of the migration is the loading of data and the building of indexes, which is different than production workloads. To analyze and optimize the use of the buffer pool, examine how buffer pools are used during the import. Figure 6-16 indicates that the temporary data is using most of the buffer pool storage. The graph shows a high amount of physical I/O resulting in efficient use of the buffer pool.

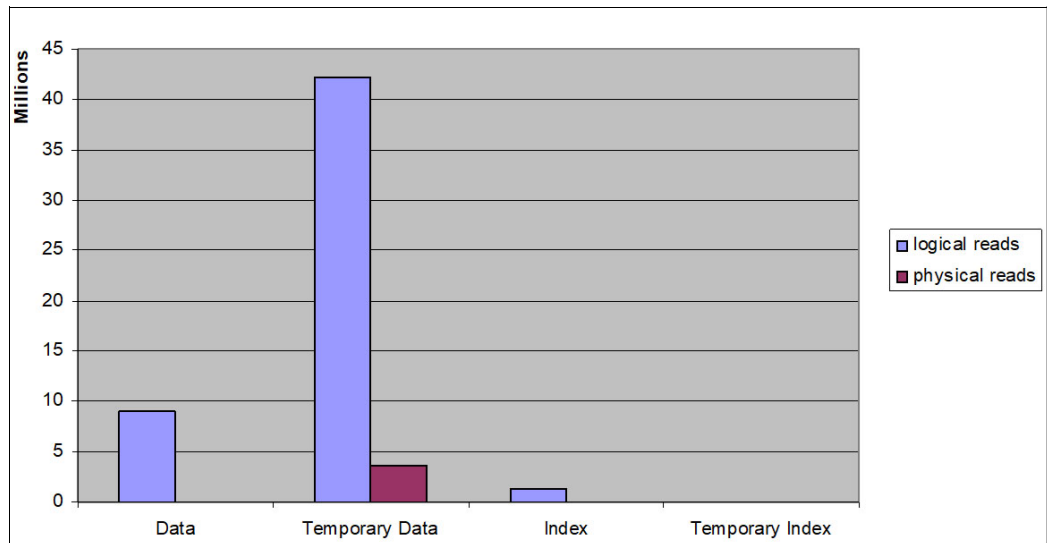


Figure 6-16 I/O Distribution

Normally you increase the buffer pool to reduce the physical I/O. However, in this situation, the temporary I/O is generated by the index build. Instead of increasing the buffer pool, optimize the sort memory areas to reduce the I/O activities to a minimum.

Note: The buffer pool configuration is of less importance for heterogeneous system copies if you use the db2load API. For memory optimization, give priority to the utility heap and the sorting memory configuration. Assign the remaining memory to the buffer pool and the other Db2 memory areas. This configuration must be changed after the system copy is completed.

6.10 SMP parallelism for index creation

The dynamic Db2 registry variable `DB2_SMP_INDEX_CREATE` overrides the default number of agents that are used to scan and sort the index data when building or rebuilding an index. During customer migrations, we have seen performance improvements to a maximum of 40% by optimizing this value. Some example timings are shown in Figure 6-17.

The Db2 registry variable increases the parallelism during index creation if CPU and I/O resources are available. The Db2 registry variable is active for all processes. Setting the Db2 registry variable to a large number might not be beneficial because it locally optimizes the index creation but blocks resources for importing the data, and it degrades performance for creating smaller indexes.

If the process of index create is the bottleneck and if you have 64 or more CPUs on the importing server, you can set the value to a higher number, for example to 16, for the entire heterogeneous system copy.

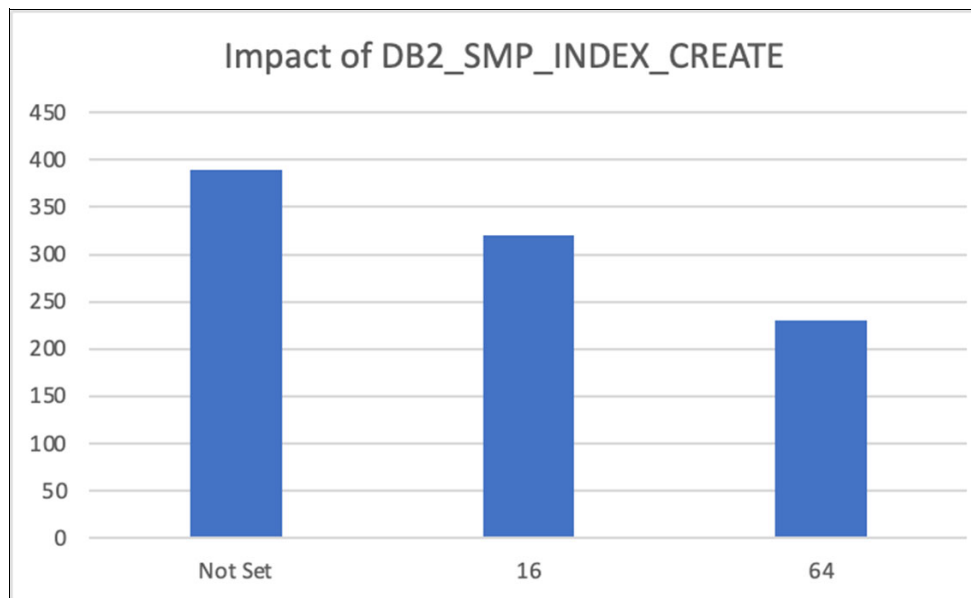


Figure 6-17 Runtime in Minutes with `DB2_SMP_INDEX_CREATE`

If the index create process of a single table or a few tables is the limiting factor, you can schedule the index create of this table at the end of the process and set the variable to a high number, such as the number of cores available.

You can achieve this by editing the relevant “.TSK” file and remove the task or set it to OK to create the index manually. Another option is to call R3load with the `-o` (omit) option to not create the indexes. See Example 6-18 on page 122 for an example file setup.

Example 6-18 Manually adapted task file for tables EDIDC

```
T EDIDC C xeq
P EDIDC~0 C xeq
D EDIDC I xeq
I EDIDC~1 C ok
I EDIDC~2 C ok
I EDIDC~3 C ok
I EDIDC~4 C ok
```

In Example 6-18 the R3load task file has entries for the table EDIDC with two optimizations:

1. The Primary Key Index is created before the data is loaded.
2. The build steps of secondary indexes are set to completed.

After the initial system copy is completed, you can set DB2_SMP_INDEX_CREATE to a high value, revert the steps for the secondary indexes to xeq, and start the R3load process again.

Because the Db2 registry variable DB2_SMP_INDEX_CREATE is dynamic, you can also interactively set the variable during the migration process. After you change the value, all new R3loads processes that trigger a CREATE INDEX statement use the new value.

Figure 6-18 shows the CPU usage of a single R3load process that creates one index with DB2_SMP_INDEX_CREATE=24.

Attention: This variable can significantly affect your migration and can effectively stop the heterogeneous system copy because it uses a significant amount of resources for this one process. In the example shown in Figure 6-18, 100% of the CPU is used and no resources are free for other tasks.

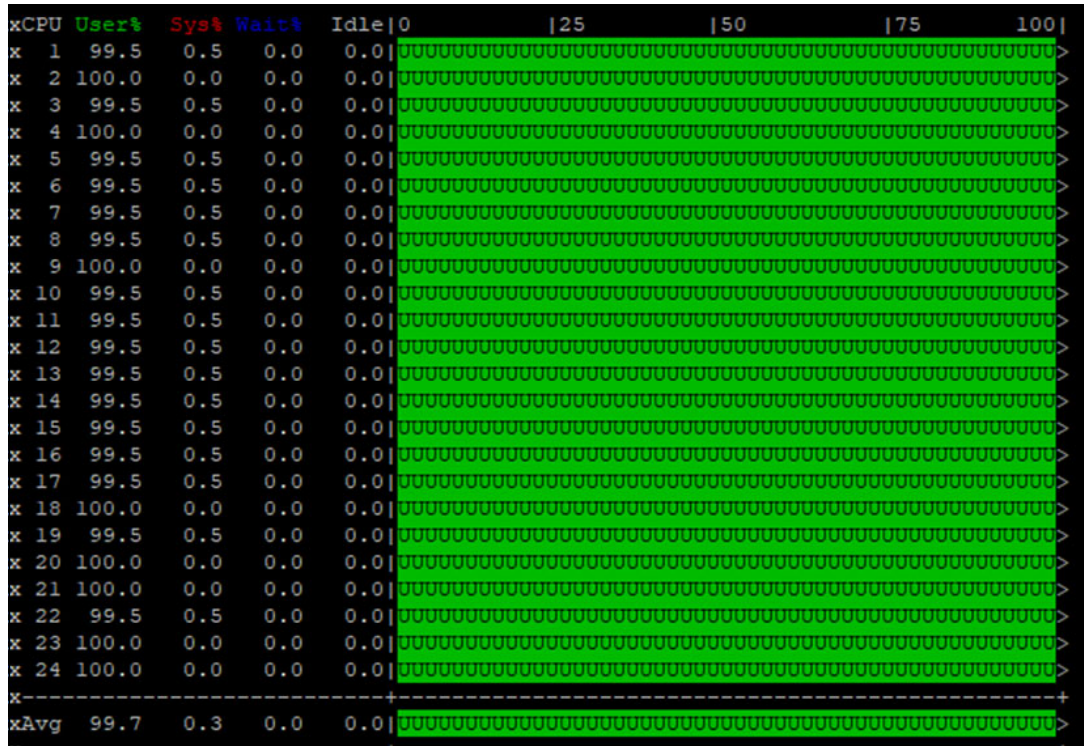


Figure 6-18 DB2_SMP_INDEX_CREATE resource usage example

6.11 LOCKTIMEOUT

During production operation you must set this parameter to an appropriate value to avoid deadlocks and to ensure that there are no applications waiting indefinitely.

During the migration process, we recommend setting the value of LOCKTIMEOUT to -1.

If you set this parameter to -1, lock timeout detection is turned off. It avoids an index creation task abort that is caused because it is waiting for a lock that is held by another long-running index creation job.

This may also be useful if you import multiple data packages in parallel using the db2load API (WHERE splitter). In this case, the first LOAD process is locked, and the other processes must wait until the data package is imported.

6.12 Optimizing statistics collection

After a database migration, the database statistics for all non-volatile tables must be collected with the RUNSTATS command to support the cost-based optimizer in generating good access plans.

For large tables this can be an expensive and time-consuming procedure. But Db2 provides some functions to improve the runtime of this step and reduces the downtime of a migration.

Db2 collects table and index statistics automatically and this is enabled by the database configuration parameter AUTO_RUNSTATS. When enabled, every 2 hours Db2 checks if statistics must be updated and performs the required statistics collection during the defined online maintenance window. Also, enable the Db2 database configuration parameter AUTO_SAMPLING so that Db2 automatically uses a page level sampling.

In addition, Db2 provides the real-time statistics (RTS) feature.

The AUTO_STMT_STATS database configuration parameter is used for RTS statistics collection. When it is enabled, the Db2 optimizer creates statistics synchronously to fabricate statistics based on other available information. The synchronous statistics collection does not use more than 5 seconds in the process and also uses sampling for larger tables. If sampling is used or the runtime exceeds the 5-second limit, Db2 schedules asynchronous statistics collection, also.

As part of the heterogeneous system copy, SAP SWPM schedules RUNSTATS jobs for a selected set of tables only and relies on the automatic or real-time statistics collection of Db2.

To obtain a recent and valid set of statistic information, the following options for optimizing the statistics collection are available:

- ▶ Enable automatic statistics collection early during the heterogeneous system copy.
- ▶ Manually run RUNSTATS.
- ▶ Extract statistics from a previous test migration using db2look and apply those using the Db2 command line processor.

6.12.1 Early enablement of Db2 automatic runstats

The SAP SWPM disables the automatic and real-time statistics collection during the import.

A possibly powerful optimization is to enable automatic statistics collection prior to or during the import.

If this is done, the automatic statistics collection starts. It comes with a slight increase on I/O and CPU resource with low single digits impact.

Other than the resource usage, there are two potential tradeoffs:

1. Some tables might be subject to multiple automatic statistics collection. This might happen, for example, if INSERT is used, and the import takes 2 or more hours.
2. The last table or the few last tables that are imported are not analyzed prior the SAP system is started. To address this, the optimizations of manual RUNSTATS or Db2 statistics copy with db2look can be used.

To enable automatic statistics collection during the import, issue the following command:

```
UPDATE DB CFG USING AUTO_RUNSTATS ON
```

6.12.2 Manual runstats

To obtain accurate statistics for the database, you can run the Db2 RUNSTATS command manually. This creates the statistics for a table manually. You can run the following SQL Statement and copy the output into an SQL Script. You can use a sampling approach to speed up statistics collection. A sample of 10 percent of all data is a good starting point. You can adapt it to your individual needs.

Example 6-19 shows the SQL statements for compiling runstats commands.

Example 6-19 SQL Statement to compile runstats commands

```
select 'RUNSTATS ON TABLE <SAP-SCHEMA>.' || tablename ||  
      ' WITH DISTRIBUTION AND DETAILED INDEXES ALL TABLESAMPLE BERNOULLI(10); '  
from syscat.tables  
where type='T' and tabschema='<SAP-SCHEMA>'  
and VOLATILE != 'C' and STATS_TIME is NULL;
```

In the SQL Statement *<SAP-SCHEMA>* must be replaced by the actual name of the SAP schema. The SQL Statement returns all non-volatile tables in the SAP schema that do not have statistics. Save this output to an SQL Script, for example `db2_stats.sql`.

You can then run the script `db2_stats.sql` with the following command:

```
db2 -tvf db2_stats.sql
```

As a further improvement, you can divide the script `db2_stats.sql` into different pieces and run the scripts in parallel.

As a best practice, you can run dedicated runstats for the largest tables that you also identified for export package or table splitting. In addition, run one or two manual runstats scripts for the remaining tables.

If enough resources are available, you can start the RUNSTATS scripts when the import for the last table is still running.

You can also use this process with the early enablement of automatic statistics collection. The SQL Statement `db2 -tvf db2_stats.sql` returns only tables that do not have statistics collected. So, at best, only a few tables remain and are used for manual statistics collection.

6.12.3 The db2look command

The following optimization can populate statistic tables on the target quickly but might not always be best for your environment. We recommend the use of the db2look command only in extraordinary situations.

After the statistical values have been calculated, they are stored in Db2 catalog tables that can be updated. Because statistics values usually exist from a previous test migration, you can extract them using the command **db2look**. Although the statistic information is outdated because it is typically based on an older snapshot of the production database, you can use it to start production on the target system. The advantage is sub-second run time for applying the runstats information about the target database that is collected during a test migration or even from the source system outside the downtime window.

Gathering RUNSTATS from the source system by using **db2look** and applying them on the target is only possible if the Db2 parameters shown in Example 6-20 are the same on the source and the target.

Example 6-20 Db2 parameters that must match to gather runstats

```
Db2 get db cfg for TAR | grep -i "Database code set"
  Database code set = UTF-8
Db2 get db cfg for TAR | grep -i "territory"
  Database territory = en_US
Db2 get db cfg for TAR | grep -i "Database collating sequence"
  Database collating sequence = IDENTITY_16BIT
```

If they do not match, subsequent Db2 automatic or real-time statistics might generate more up-to-date statistics, which would be incorrect.

Attention: After table or index statistics are updated by using **db2look**, the table is excluded from automatic or real-time statistics collection.

To re-enable automatic runstats after the import of the statistics with db2look, run a manual RUNSTATS. To re-enable automatic and real-time statistics collection use the following command for each corresponding table:

```
RUNSTATS ON TABLE <tablename> SET PROFILE NONE
```

This RUNSTATS command can be run while the SAP System is online and thus outside the downtime windows. Statistics will not be collected automatically until the command is run.

Example 6-21 on page 126 shows how to generate a **db2look** SQL file for the update of table EDIDS in database NZW with schema name sapnzw. The name of the generated file is edids_db2look_stats.sql.

Example 6-21 db2look command to extract statistics for a table

```
db2look -d nzw
        -m -r -c
        -z sapnzw -t edids
        -o edids_db2look_stats.sql
```

Note: When you run the **db2look** command, you must include the **option -r**. Otherwise, the generated script contains RUNSTATS commands without any performance advantage.

Also, you must apply the extracted statistics information to the Db2 target system as part of the migration. You can use the following command:

```
Db2 -tvf <tablename>_db2look_stats.sql
```

You can combine the db2look procedure with the early enablement of automatic statistics collection and the manual statistics collection and use the db2look-based procedure only for the last few remaining tables.

An optimized procedure can include the following steps:

1. Enable automatic statistic collection during the import.
2. Generate and run the manual RUNSTATS commands shortly after all tables are created and before the last import ends.
3. Exclude the last remaining table or extremely large table from the RUNSTATS script and apply previously collected statistics that are extracted with db2look.
4. Enable automatic runstats for this table afterward during uptime of the target system.

Often, the combination of early automatic statistics collection and manual RUNSTATS is sufficient. We recommend the db2look procedure only in extraordinary situations.

6.13 Importing by use of a database server versus a remote client

You can often optimize the database export by using application servers because R3load itself uses a significant portion of the available CPU resources. This applies when codepage conversion or declustering is done during the export.

Usually, the benefit of using application servers for the import is not as significant as during the export.

Figure 6-19 shows the result of tests with local and remote R3load processes. It shows the workload distribution, based on AIX WLM monitoring, between database and R3load gathered from a migration of a customer system.

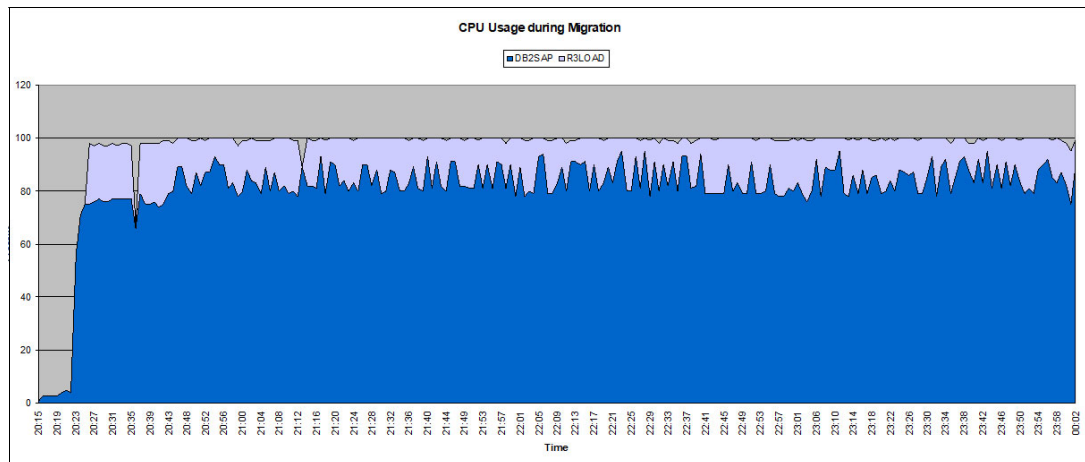


Figure 6-19 Workload distribution – Database versus R3load

Based on these results, a rule of thumb is that about 10%–15% of the CPU workload for the import can be shifted away from the database server by using a separate application server for R3load. However, the database load was not faster in our test case when using remote R3load because of the significant increase in network traffic as shown in Figure 6-20. The data between the SAP application server and the database server is not compressed, so the process loses the advantage of shipping less data over the network.

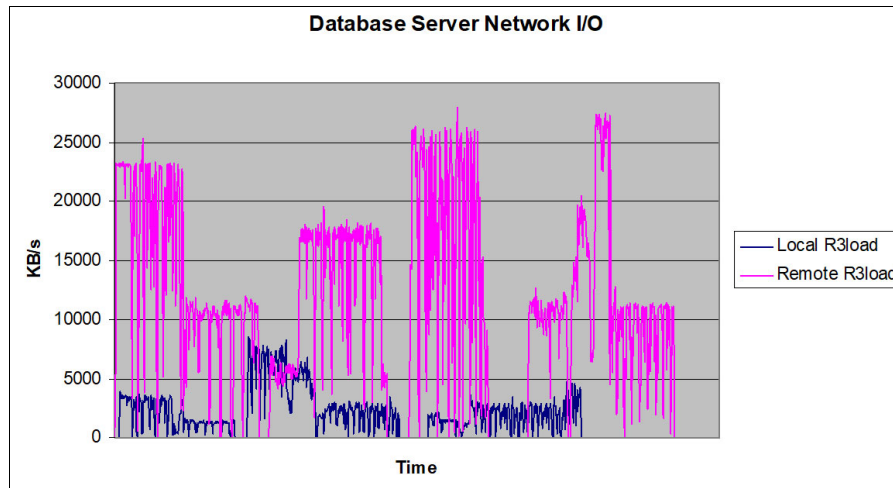


Figure 6-20 Network I/O during Import with local and remote R3load

6.14 Other Db2 features

When you consider the process and resources used during the import, there might be other potential areas of optimizations. In our tests and during projects with customers, we found that the Db2 configuration used during normal production also works well during the import. There are some areas that might improve the import, but the benefit depends on the system

environment Therefore, some of the features might help provide a slight increase in import throughput, but we cannot give a general recommendation to use those features.

6.14.1 Using self-tuning memory management

Db2 self-tuning memory management (STMM) balances memory usage of the database memory users based on the requirements and costs. These costs are associated with changing the amount of memory for each memory user.

STMM decreases the need for detailed memory configuration. It can tune the memory usage from the Db2 database global memory for the following Db2 memory users:

- Database **LOCKLIST** and **MAXLOCKS**
- Package cache size (**PCKCACHESZ**)
- Sort memory (**SHEAPTHRES_SHR**, **SORTHEAP**)
- Buffer pools

The values of LOCKLIST, MAXLOCKS and PCKCACHESZ do not play a significant role in the performance of the heterogeneous system copy, but the configuration for sorting and buffer pools is important. The sorting configuration is more important because the Db2 LOAD utility loads data into the database and bypasses the buffer pools. However, the buffer pool is still important. Therefore, STMM can help to find the optimal configuration.

Figure 6-21 shows the configuration changes and runtime improvements for five migrations in a row.

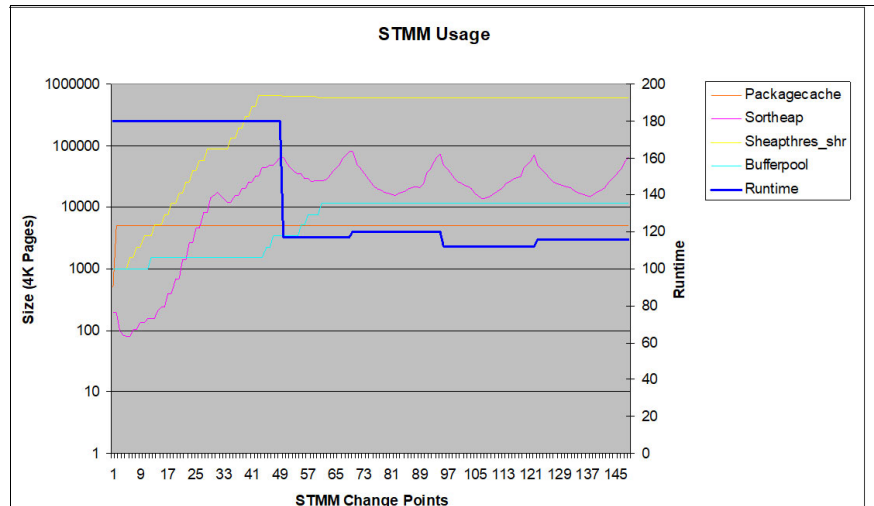


Figure 6-21 STMM Configuration Changes

During the five test migrations, STMM changed the Memory configuration about 150 times, and we have called this action an STMM Change Point.

Each migration run starts at the following STMM Change Points:

1. Migration at STMM Change Point 1
2. Migration at STMM Change Point 49
3. Migration at STMM Change Point 69
4. Migration at STMM Change Point 95
5. Migration at STMM Change Point 121

STMM performed the configuration changes for the buffer pool, package cache and the Db2 sort configuration. The sizes for **MAXLOCKS** and **LOCKLIST** were unchanged during the tests and are not part of the figure. The primary axis on the left displays the memory size for each of the monitored memory settings and the secondary axis, the runtime of the import is shown.

You can see in Figure 6-21 on page 128 that STMM adapts the size for the **SORTHEAP** and **SHEAPTHRES_SHR** more quickly than the increases of the buffer pool. At position 50, the first test migration was completed, and the second test migration was already close to the optimal runtime.

The improvement from Db2 STMM occurred in the **SORTHEAP** and **SHEAPTHRES_SHR** configuration, which improved creating indexes.

The chart also shows that the last test migration (Position 120-150) was slightly slower compared to the previous test run. This might happen but is not significant. More significant is the fact that it changes at all, and this conflicts with the best practice to conduct the final test migration with the same settings and schedule as the last successful test migration.

So, based on the tests, we can draw the following conclusions:

- ▶ Db2 STMM adapts to the special workload during a heterogeneous system copy. To support STMM, you can specify the memory consumptions with starting values and let STMM adjust the values.
- ▶ Start with the following values:
 - SORTHEAP of 100,000
 - SHEAPTHRES_SHR=SORTHEAP times the Number of parallel R3load processes
 - Assign most of the remaining memory to the buffer pool.
- ▶ To verify a consistent behavior, disable STMM during the final migration test.

6.14.2 INTRA_PARALLEL

This parameter specifies whether the database manager can use intra-partition parallelism and can be set in accordance with *SAP Note 2047006 – DB6: Use of Db2 SMP Parallelism (INTRA_PARALLEL=YES)*.

In our tests, we do not see any noticeable positive or negative performance impact. In theory, the parameter might help if the R3load Option **SPLITTED_LOAD** is used. It can improve the final copy phase, but typically, the bottleneck is more I/O related, so the Db2 parallel processing does not help much.

6.14.3 Changed pages threshold and alternate page cleaning.

You can use the changed pages threshold (**CHNGPGS_THRESH**) parameter to specify the percentage of changed pages at which the asynchronous page cleaners are started if they are not currently active. If the page cleaners are started, they build a list of the pages to write to disk. After they finish writing those pages to disk, the page cleaners become inactive again and wait for the next trigger to start. The **CHNGPGS_THRESH** parameter is important when large tables are imported by **INSERT**, such as parallel insert of split tables. Because all the pages in the buffer pool must be flushed to disk after a commit, this operation might slow throughput because a large portion of data is written to disk in one operation. If you set **CHNGPGS_TRESH** to a lower value, the dirty pages can be written earlier, and this might speed up the import process.

Conversely, decreasing this value might affect the performance of index build. The CHNGPGS_TRESH parameter can force writes of temporary data for index creation to disk even if it fits into the buffer pool.

Tuning this parameter is difficult, and the tests did provide any general recommendations, so for most cases, use the default value of 20.

6.14.4 Alternate page cleaning

You can configure page cleaners on your system to be proactive with the registry variable DB2_USE_ALTERNATE_PAGE_CLEANING.

When you set the registry variable to ON, page cleaners behave more proactively in choosing which dirty pages are written to disk. After enablement, page cleaners no longer respond to the value of the CHNGPGS_THRESH database configuration parameter.

During our tests, we did not see a clear positive or negative impact, so we cannot give a recommendation whether to use it.

6.14.5 Spilled sorts to a RAM drive

The optimal configuration of SORTHEAP and SHEAPTRES_SHR can reduce the I/O on temporary tablespaces. If you cannot reduce the number or amount of data that is spilled to disk, you can optimize the layout of the temporary tablespace as described in the Db2 layout section in this document.

One method to optimize the temporary I/O is to use a RAM drive and create the temporary tablespaces in this memory area. The advantage is fast I/O

The results may be promising at the first, but with a decent configuration of SORTHEAP and SHEAPTRES_SHR, we did not see a clear benefit of this concept. If you have enough main memory on the target machine, configure Db2 with a large SORTHEAP and SHEAPTRES_SHR instead of creating a RAM drive.

6.14.6 Disabling database logging

You can use the `-nolog` parameter with R3load to avoid logging of data when it is inserted into the database. With this R3load option, you can significantly reduce logging. However, best practice is to use the db2load API. In this case, the `-nolog` option affects only small tables or exceptional tables that are not using the db2load API.

You can deactivate logging during R3load, which might avoid disk contention and logging overhead during inserts. Also, if you run the roll forward utility and it encounters a log record that indicates that a table in the database was populated with the NOT LOGGED INITIALLY option, the table is marked as unavailable. Therefore, perform a full offline backup after the migration to ensure that the database and all tables are recoverable.

The `-nolog` option reduces the amount of logging but is typically not faster compared to logging enabled. So, use this option only when logging I/O is limiting the performance.

Important: Do not use the `-nolog` parameter of R3load when using parallel import of table splits with INSERT. This creates lock wait situations and sequential processing of tables.

6.14.7 Db2 native encryption

If you are moving to the cloud with the IaaS model, you can encrypt the data on disk. Db2 offers the feature of native encryption that encrypts data on disk, such as Db2 tablespace container files and transaction log files.

With native database encryption, the database system itself encrypts the data before it calls the underlying file system to write that data to disk. This means not only your current data is protected, but also data in new tablespace containers or tablespaces that you might add in the future. Native database encryption is suitable for protecting data in cases of either physical theft of disk devices or privileged user abuse.

Native database encryption also ensures that damaged or outdated disks that are replaced and scrapped by the cloud service provider do not contain readable data.

However, Db2 native encryption introduces an additional processing step in all I/O operations. This is the decryption process when data is read from disk or the encryption process when data is written to tablespaces or the Db2 transaction log files.

The encryption process is supported by hardware acceleration starting with Db2 11.1. Hardware acceleration makes a significant difference in the impact on both system resource consumption and application throughput. Db2 automatically uses the following CPU enhancements:

- Intel Advanced Encryption Standard New Instructions (AES-NI) support
- IBM Power8® in-core support for the AES

The hardware acceleration is automatically detected and used by Db2. To determine whether your systems are capable of this, check the Db2 diagnostic log file for the following entry:

Encryption hardware acceleration detected

Note: Most of the SAP certified servers offered by the major cloud service providers provide AES-NI support for Linux and Windows, and you should use Db2 native encryption only with available support for this feature.

The effect of the additional processing needed for encryption and decryption is slower disk access. If you enable Db2 native encryption, monitor the following Db2 metrics closely before and after the enablement:

pool_read_time	Indicates the total amount of time spent reading in data and index pages from the tablespace containers (physical) for all types of tablespaces.
pool_write_time	Cumulative elapsed time for each asynchronous write to the tablespace containers to complete
log_disk_wait_time	The amount of time an agent spends waiting for log records to be flushed to disk.

There are several options to enable Db2 native encryption. It can be enabled during a restore operation, which can be optimized with HADR to minimize downtime. It can also be enabled during the database creation time, which is an option in the SAP SWPM. The database can be created with Db2 native encryption enabled, and all data that is subsequently imported by R3load is encrypted.

Because Db2 native encryption introduces additional processing and additional resource usage, it might slow down the migration process. Although we see minimal impact during the

export on an IBM Power Server, the impact during the import is more significant. This is shown in Figure 6-22.

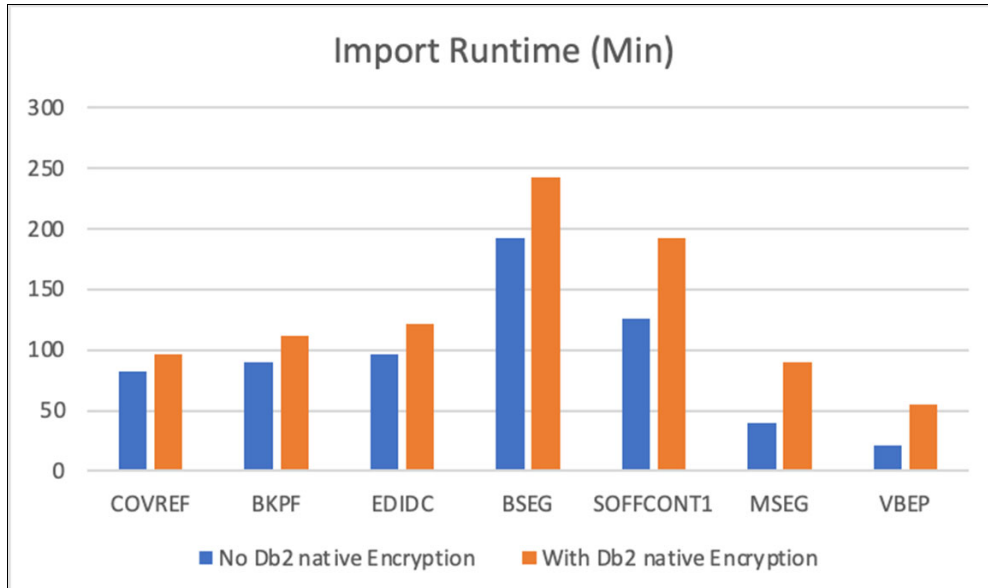


Figure 6-22 R3load Import with and without Db2 native encryption enabled.

The graph shows significant impact on some tables. The import time, including the index build for the tables MSEG and VBEP are doubled, and the impact on other tables is a performance degradation of 20%–50%. The impact depends on the table structure but even more on the number and width of the indexes defined.

Therefore, the overall impact on the migration process can vary, depending on the largest tables in the system. However, the impact is noticeable, so plan for an additional 50% of runtime of the heterogeneous system copy with Db2 native encryption enabled.

Other than the default optimization to minimize I/O by reducing spilled sorts and the `-nolog` option for tables that are using insert, there is not much optimization potential for Db2 native encryption.

Assess if the heterogeneous system copy still fits in the downtime window or if you want to implement the feature later by using backup and restore, perhaps downtime optimized by using HADR. For details about this process optimization with HADR, see the [Db2 for SAP Community page](#).



Import/Export overlap

This chapter includes discussions of socket transfer and table splitting. These options require specific settings on both the export and the import side.

This chapter describes some important facts about both features to provide you with a better understanding of how they can impact your migration.

The following topics are discussed in this chapter:

- ▶ 7.1, “Socket transfer and table split overview” on page 134
- ▶ 7.2, “Socket option” on page 136
- ▶ 7.3, “Table splitting” on page 139

7.1 Socket transfer and table split overview

The motivation for socket transfer is because you do not want to wait to start the import until the export is completed. You want to start the import immediately when the export starts. See Figure 7-1. With this, you can optimize the scheduling of the export and import and possibly save 50% of the time.

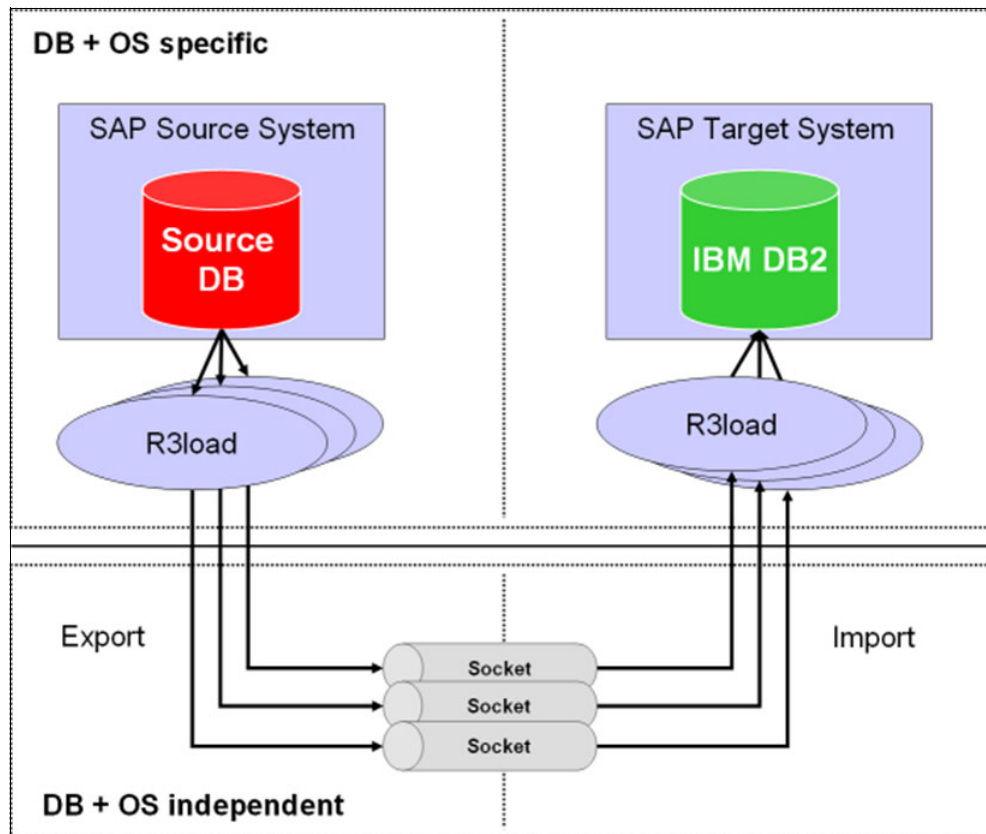


Figure 7-1 The R3load Socket Option

By using table split, multiple goals can be achieved. The export can be improved by running multiple R3load processes that export a single table in distinct chunks or splits. The improvement for this comes to some extent from the database processing. Db2 is typically able to deliver data much faster than a single R3load can receive, compress, and write to disk.

The benefit of multiple R3load exports against a single table comes primarily from the effect of parallelization of the R3load internal processing. This is even more true if a code-page conversion or declustering is done on the export side.

When running an import, improvements are possible as well, but knowing the differences between how Db2 handles LOAD versus INSERT processing is important to understand the different import optimizations. This is presented in 6.6.1, “Db2 LOAD versus INSERT” on page 107.

You can run only one R3load, which uses the db2load API per table. So, you might have an export running with multiple R3load processes, and the import sequentially loads the export dumps. Although this seems to be a disadvantage, it can also be used to achieve similar

effects as the socket transfer option, which is an overlapping of export and import as shown in Figure 7-2

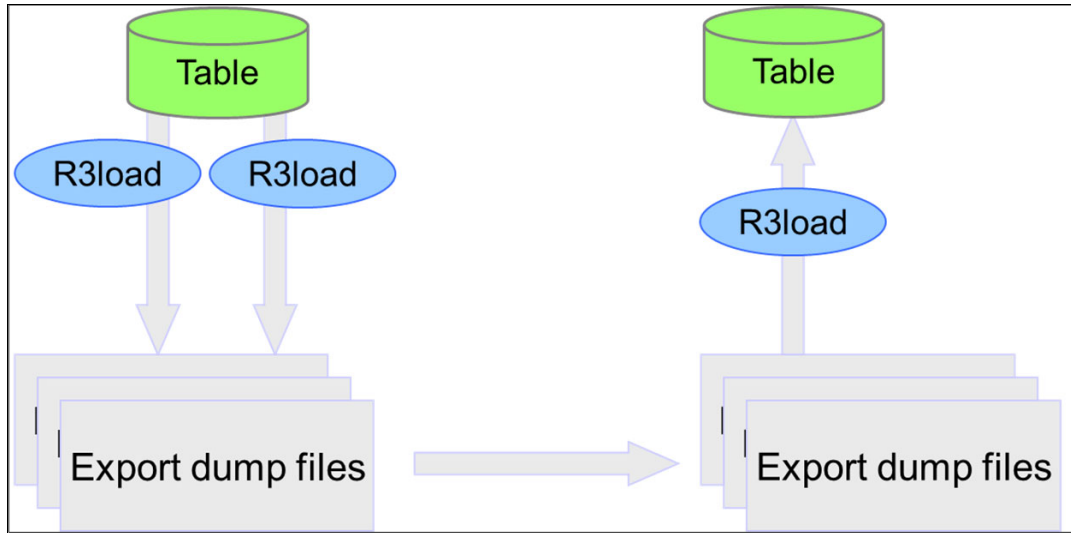


Figure 7-2 Table Split with single R3load using Db2 LOAD: "Forced Load Option"

To accomplish this overlap, you define a reasonable number of splits and do not run them in parallel but sequentially. Then transfer the files to the target by configuring the network or FTP exchange directory and let the export monitor and the import monitor communicate using the signal files to manage the import process.

With this, you achieve almost the same effect as with the socket option. Multiple chunks are defined but only one R3load is exporting at a time and on the target, the import is managed so that only one R3load using the db2load API is running at a time. This concept is called *Forced LOAD*, and the effect is almost the same as the socket transfer option. The difference is that the importing R3load must wait for the first table split to complete the export.

Using INSERT is typically slower than Db2 LOAD, but because multiple R3load processes with INSERT can run in parallel, another option is to use multiple R3load streams for both import and export as shown in Figure 7-3.

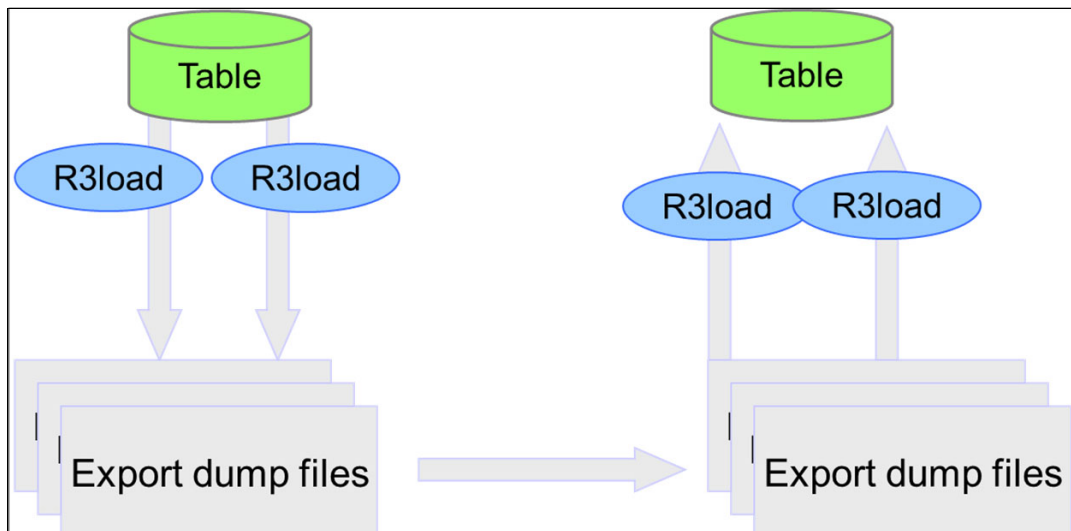


Figure 7-3 Table Split with multiple R3load processes using INSERT

With multiple parallel R3load processes for one table on the importing side, you also parallelize the R3load internal processing, such as decompressing the R3load dump files, which can provide an efficient use of resources and reduce the migration time.

The third option is to import the data into multiple temporary tables with the db2load API, followed by a final database internal copy with LOAD FROM CURSOR into the final table. This option is called the *Split Load* option and is shown in Figure 7-4

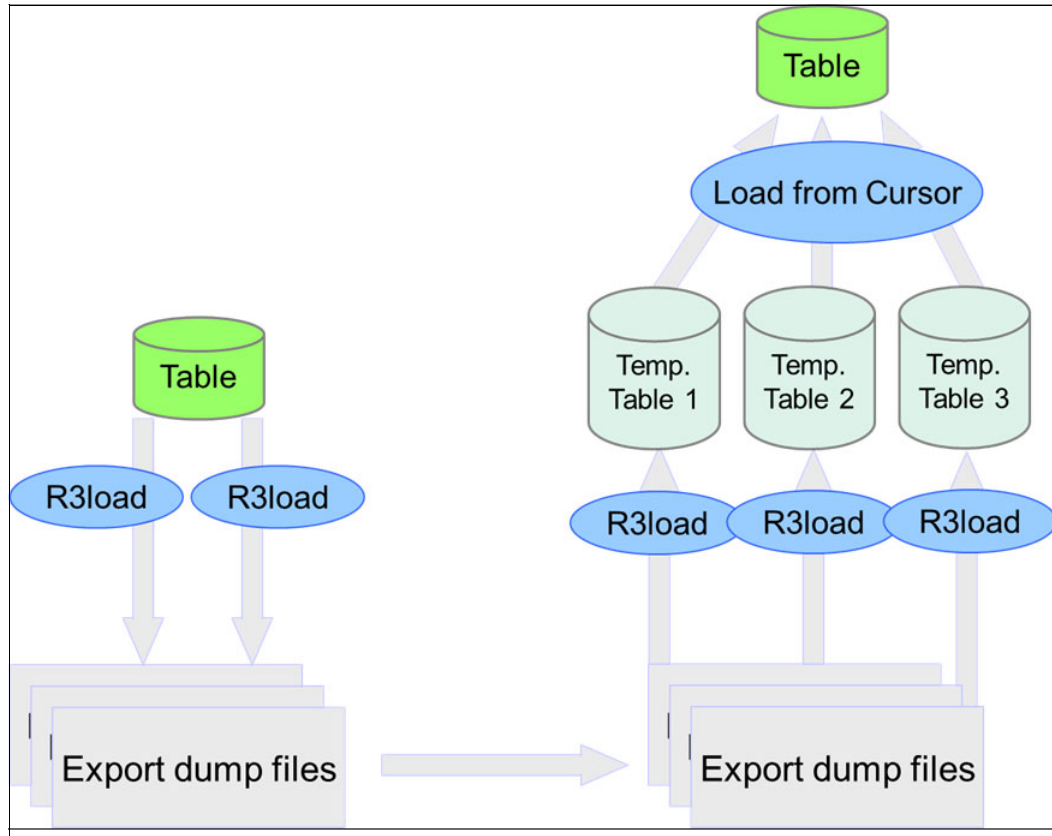


Figure 7-4 Table Split with single R3load using Db2 LOAD: “Split Load Option

Although the data is loaded twice, the procedure can be much faster than just one R3load. The final LOAD FROM CURSOR does not involve any R3load processing or Db2 internal code page conversion (UTF-16 to UTF-8) and is typically by factors faster than R3load with using the db2load API.

7.2 Socket option

Data that is exported from the source system can be transported to the target system by using various methods. You can, for example, store the data on disk and transport it to the target server using a transportable device (tape, disk), a network share or FTP. The network sharing or FTP transfer can be accompanied with tools that automate and optimize the transfer.

7.2.1 Comparing file and socket migration method

For this example, the table BKPF was exported and the results are shown in Figure 7-5.

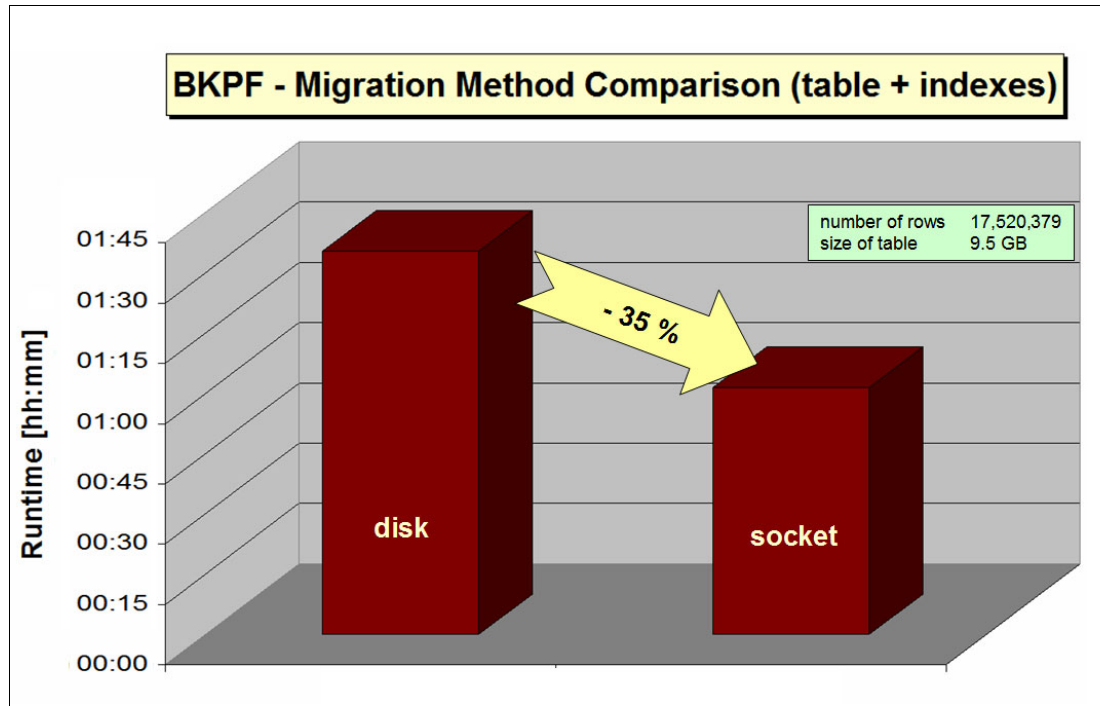


Figure 7-5 Migration Method Comparison (Disk Dump versus Socket)

In our test environment, we measured a reduction of the runtime by 35%. This can lead to a substantial reduction of downtime during a heterogeneous system copy. However, the best expected outcome of a 50% reduction was not achieved.

The best result that you can achieve is when the export and import processes run with the same throughput. When one part is slower, it determines the overall improvement. In the test case with the table BKPF, the export was the limiting factor.

7.2.2 Migration Monitor (MigMon) configuration

You can configure the migration monitor (MigMon) to use the sockets feature for a unicode conversion as shown in Figure 7-6 on page 138.

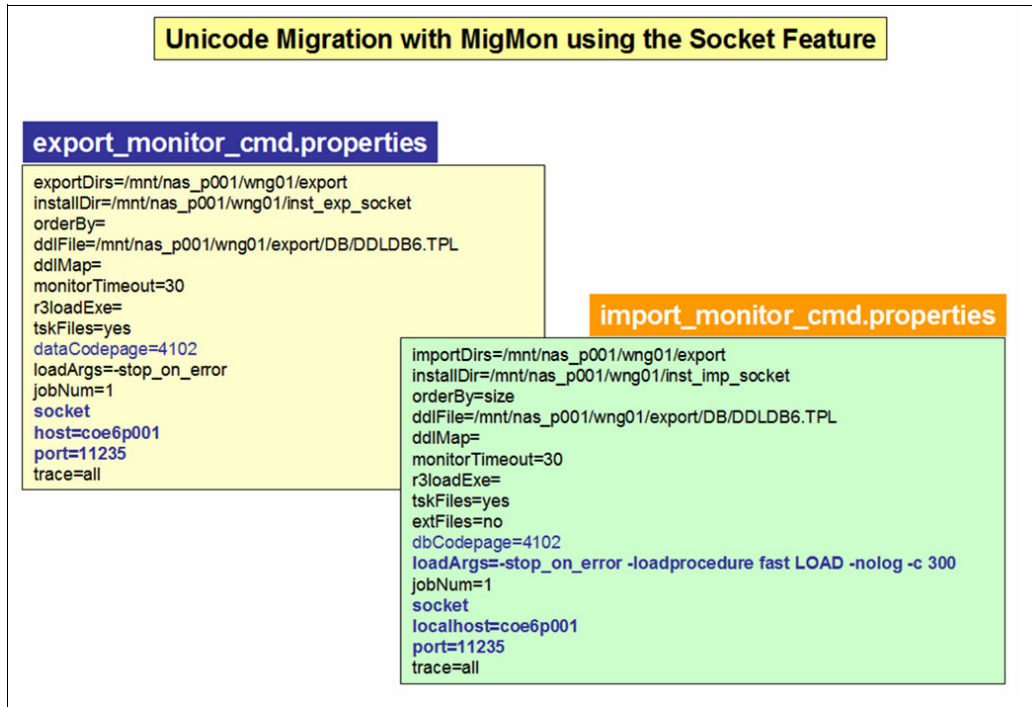


Figure 7-6 MigMon Configuration for Unicode Conversion with Socket Feature

To configure the socket option, you must set the parameters in the MigMon properties file as shown in Table 7-1.

Table 7-1 Migration Monitor Socket Options

Option	Description	Comments
socket	Socket operating mode	R3load does not write dump files to the file system, but the export and import work through the socket connection.
host	Remote import host	Name or IP address of the import host specified on the export host
port	Host port number	Must be the same as the port number on the import host. Any free port on the import host from 1024 to 65535.

The port definition on the export and import sides must match. Otherwise, there is no connection between them.

You can also use this option within a server. A possible implementation scenario is to use the importing server also as an application server for the export and to define the socket communication locally. If you want to combine this option with other techniques of data transport, you must use multiple MigMon instances because of different configurations in the properties file.

You can also use the socket option for Unicode conversions if you adhere to the restrictions described in *SAP Note 971646 - 6.40 Patch Collection Hom./Het.System Copy ABAP*.

7.2.3 Socket transfer considerations

The socket transfer is an option that saves disk space and I/O because it does not materialize the exported data on disk. However, the option includes some tradeoffs. If the tradeoffs are not applicable to your environment, the socket transfer option can be a good choice.

You must have a stable network connection between the export and import sides. If there is an error in the data stream, the process fails and must be restarted. This is especially true if you have a R3load pair that handles a large table.

Restart of a failed socket transfer can be more complex or might take more time. In general, if one process of an R3load pair fails, the remaining processes must also be stopped. For various combinations of restarting a socket R3load, refer to the *SAP Heterogeneous System Copy Guide*.

When using the socket option, declustering must occur on the target server, and the migration monitor must be set up differently. Set `SUPPORT_DECLUSTERING=false` on the export side and `SUPPORT_DECLUSTERING=true` on the import side. This might seem like a small change, but it deviates from the SAP default of declustering during the export.

Socket transfer does not compress the data stream. Usually, the R3load export dump files are compressed. Although this is a CPU-intensive operation, significantly less data is transferred over the network. With the socket transfer, dump data is not being compressed, so more data is shipped over the network. This might be an issue for heterogeneous system copies from on-premises into the cloud. On the other side, the socket transfer decreases the CPU resource usage for R3load compression.

The Db2 specific R3load option `OPT_COMPRESS` cannot be used with socket transfers. You can use only the automatic dictionary creation by Db2 or the `FULL_COMPRESS` option that reorganizes the table after it is loaded. The various compression options are presented in section 6.5.1, “Introduction to Db2 compression” on page 96.

7.3 Table splitting

The contents of a single table can be exported and imported with multiple R3load processes in parallel. This table splitting approach is typically used for large tables to improve the export and import runtime.

Important: Table splitting is an advanced option, and three different methods are available for importing split tables with a target Db2 database: Forced LOAD, parallel insert, and split LOAD. The decision on which type of import to use depends on the table structure and infrastructure and requires testing.

Also keep the number of splits significantly less than the maximum of 200.

Consider 10–20 splits per table or a maximum of 100 GB of data volume per table split as a starting point.

7.3.1 Table splitting introduction

To be able to export a single table with multiple parallel R3load processes, each process requires a WHERE-file that specifies a distinct range of records in the source table. This set of records is exported into a dedicated set of dump files by an R3load process. You can

create the WHERE files manually; with the available tools R3ta or SAPuptool; by using SAP SWPM; or during a manual Migration Monitor process.

The two key parts that are used are the following configuration files:

- ▶ *.STR files, which are also used with the standard migration process
- ▶ *.WHR files, each is a dedicated file per split that contains the WHERE clause for the table split

This is shown in Example 7-1.

*Example 7-1 *.STR and *.WHR Files for table COVREF that is divided into 10 splits.*

```
bash-5.1$ ls -ltr *COVR*
rwxrwxrwx  1 srcadm  sapsys          1172 Mar 16 2022 COVREF.STR
rwxrwxrwx  1 srcadm  sapsys           298 Jun 16 2022 COVREF-9.WHR
rwxrwxrwx  1 srcadm  sapsys           310 Jun 16 2022 COVREF-8.WHR
rwxrwxrwx  1 srcadm  sapsys           316 Jun 16 2022 COVREF-7.WHR
rwxrwxrwx  1 srcadm  sapsys           322 Jun 16 2022 COVREF-6.WHR
rwxrwxrwx  1 srcadm  sapsys           322 Jun 16 2022 COVREF-5.WHR
rwxrwxrwx  1 srcadm  sapsys           322 Jun 16 2022 COVREF-4.WHR
rwxrwxrwx  1 srcadm  sapsys           322 Jun 16 2022 COVREF-3.WHR
rwxrwxrwx  1 srcadm  sapsys           320 Jun 16 2022 COVREF-2.WHR
rwxrwxrwx  1 srcadm  sapsys           147 Jun 16 2022 COVREF-10.WHR
rwxrwxrwx  1 srcadm  sapsys           165 Jun 16 2022 COVREF-1.WHR
```

The *.WHR files contain the WHERE clause of the SQL Statement that is used by R3load to export this chunk of the table. The two SQL statement parts each have different values for the columns RELID, PROGRAMME and SRTF in Example 7-2 for the table COVREF.

Example 7-2 Content of two different WHR-Files for the table COVREF

```
tab: COVREF
WHERE ('GK' < "RELID" OR ("RELID" = 'GK' AND '%ARBERP_H_ONSTMSA' < "PROGRAMME")
OR ("RELID" = 'GK' AND "PROGRAMME" = '%ARBERP_H_ONSTMSA' AND 0 < "SRTF2")) AND
("RELID" < 'IC' OR ("RELID" = 'IC' AND "PROGRAMME" < '%ARBFND_H_MSGCSTA') OR
("RELID" = 'IC' AND "PROGRAMME" = '%ARBFND_H_MSGCSTA' AND "SRTF2" <= 0))
tab: COVREF
WHERE ('FE' < "RELID" OR ("RELID" = 'FE' AND '%ARBERP_H_CXMLMSG' < "PROGRAMME")
OR ("RELID" = 'FE' AND "PROGRAMME" = '%ARBERP_H_CXMLMSG' AND 0 < "SRTF2")) AND
("RELID" < 'GK' OR ("RELID" = 'GK' AND "PROGRAMME" < '%ARBERP_H_ONSTMSA') OR
("RELID" = 'GK' AND "PROGRAMME" = '%ARBERP_H_ONSTMSA' AND "SRTF2" <= 0))
```

There are many options available to create and optimize the table split migration. Therefore, we focus on Db2 specific information only. For details about the table split, refer to the SAP heterogeneous system copy guide.

7.3.2 Export considerations

The SQL statement used by R3load to export a table looks like the following:

```
SELECT * FROM <SCHEMANAME>.<TABLENAME> ORDER BY <ORDERCOLS>
```

The order by is appended if the DDLDB6.TPL file contains the ORDER_BY keyword or if the table needs to be exported as a sorted table. For example, if a table cluster is exported and a code-page conversion is required, a sorted export is performed. The ORDERCOLS are the columns defined by the primary key on the table.

If table split is used, the statement defined in the WHR-File is merged into the general statement. The actual statement looks similar to the statement in Example 7-3.

Example 7-3 Simplified SQL Statement for sorted export with table split

```
SELECT * FROM SAPSRC.COVREF WHERE ('FE' < "RELID" OR ("RELID" = 'FE' AND
'%ARBERP_H_CXMLMSG' < "PROGNAME") OR ("RELID" = 'FE' AND "PROGNAME" =
'%ARBERP_H_CXMLMSG' AND 0 < "SRTF2")) AND ("RELID" < 'GK' OR ("RELID" = 'GK'
AND "PROGNAME" < '%ARBERP_H_ONSTMSA') OR ("RELID" = 'GK' AND "PROGNAME" =
'%ARBERP_H_ONSTMSA' AND "SRTF2" <= 0)) ORDER BY RELID, PROGNAME, SRTF2
```

Example 7-3 shows the additional optimization options that are related to SQL tuning R3load scheduling.

Accurate scheduling and the number of R3load processes are also an important part of optimization.

Table splitting typically uses the order-by.txt file to define a dedicated schedule for large tables. When you make a decision on the import strategy, consider that the number of R3load processes on the source system must be defined in coordination with the import process.

If you choose to use a sequential import using the db2load API, it might not be beneficial to have many concurrent R3load exports. You might consider reducing them to a sequential single R3load export, or a few exporting R3loads if a single export is slower than a single import.

If you decide to import by using temporary tables, balance the number of export and import tasks. Consider that the exporting R3load tasks are typically slightly faster than the importing R3load tasks.

Note: You can define the number of concurrent R3load jobs in the Migration Monitor properties file and the order-by.txt file. If you do so, the total number is the sum of the jobs in both files.

As an example, if you set jobNum=5 in the export_monitor_cmd.properties file and use the order-by.txt file shown in Example 7-4, then a maximum of nine R3load export processes are run in parallel.

Example 7-4 Total Number of R3load processes

```
[FAGLFLEXA]
loadArgs=-dbcodepage 4103
jobNum=2
FAGLFLEXA-1
FAGLFLEXA-2
FAGLFLEXA-3
FAGLFLEXA-4
FAGLFLEXA-5
FAGLFLEXA
[COVREF]
loadArgs=-dbcodepage 4103
jobNum=2
COVREF-1
COVREF-2
COVREF-3
COVREF-4
```

There are more options available to distinguish between large and small packages, and you can find details in the *SAP Heterogeneous System Copy* guide.

To summarize, be aware of the number of parallel R3load processes, define them and incorporate both the export and import side in the decision to ensure that no single resource is overused.

Figure 7-7 shows an export where all the available CPU resources are consumed solely by the split export of the table CDCLS. For the cost of optimizing the table CDCLS, the machine had no free resources of the remaining export. The picture also contains the workload separation between R3load processing and database processing and demonstrates that Db2 is not very busy.

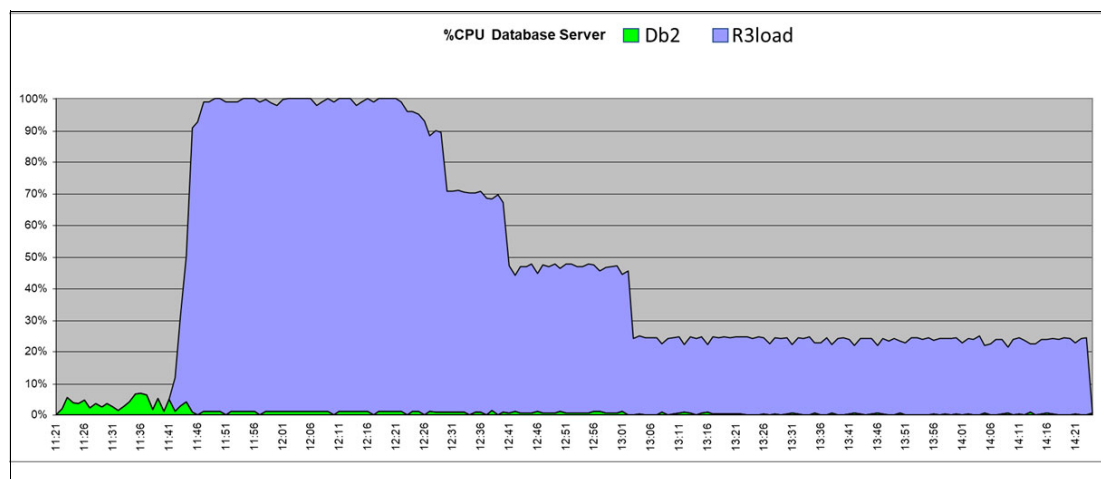


Figure 7-7 CDCLS Export – Parallelism 8 – CPU Consumption

Create temporary index

To accelerate the export, an index on the split tables can be created with R3load using the .cmd and .TSK files that were generated by R3ta. Creating additional indexes on the production system can affect the workload on the system and might also result in different access plans for the queries that are part of the production workload. Therefore, this kind of export optimization should be carefully configured and closely monitored.

Reorganize tables

When you experience a slow export on a split table, check the cluster factor of the index using the DBA Cockpit in your source system.

If you export unsorted tables, a low cluster factor can significantly reduce the performance of the data unload. To resolve the performance issue, you might reorganize the table by using the index before you start the export. Reorganizing the table increases the workload on the system.

In tests, reorganizing GLPCA according to the index used for the export, resulted in a significant runtime reduction. Export performance improved by a maximum factor of 6.7 as shown in Figure 7-8 on page 143.

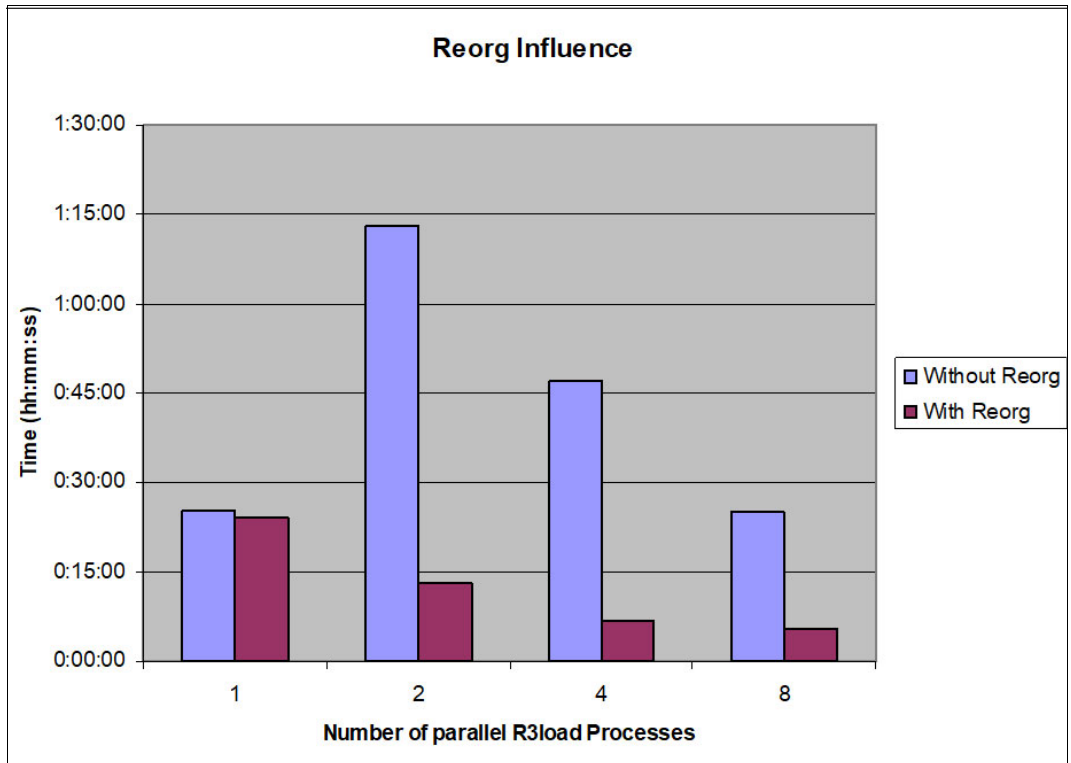


Figure 7-8 REORG Influence on Export Time

Another important observation from Figure 7-8 is that the export time decreases almost linearly when increasing the number of parallel jobs from 1 to 2 and then to 4 parallel jobs. When increasing from 4 to 8 parallel jobs, the performance gain was only factor 1.3. With 8 parallel jobs the CPU usage was 30% and I/O wait time was at most 50%. Therefore, we conclude that in the test case with 8 parallel R3load jobs, the system performance was bound by the limited I/O capacity of the storage.

Non-optimal access plan

If you look at the different WHR-Files, the SQL Statement in the first and last WHR-File is different from the other files. In all other WHR-Files, the statement describes a specific range with a clear upper and lower limit. The first WHR-File includes everything below a certain limit, and the last file includes everything above a certain limit. Depending on the criteria used, the Db2 optimizer might choose a non-optimal access plan and thus the first and last table split export might be significantly slower. This is demonstrated in Figure 7-9 on page 144.

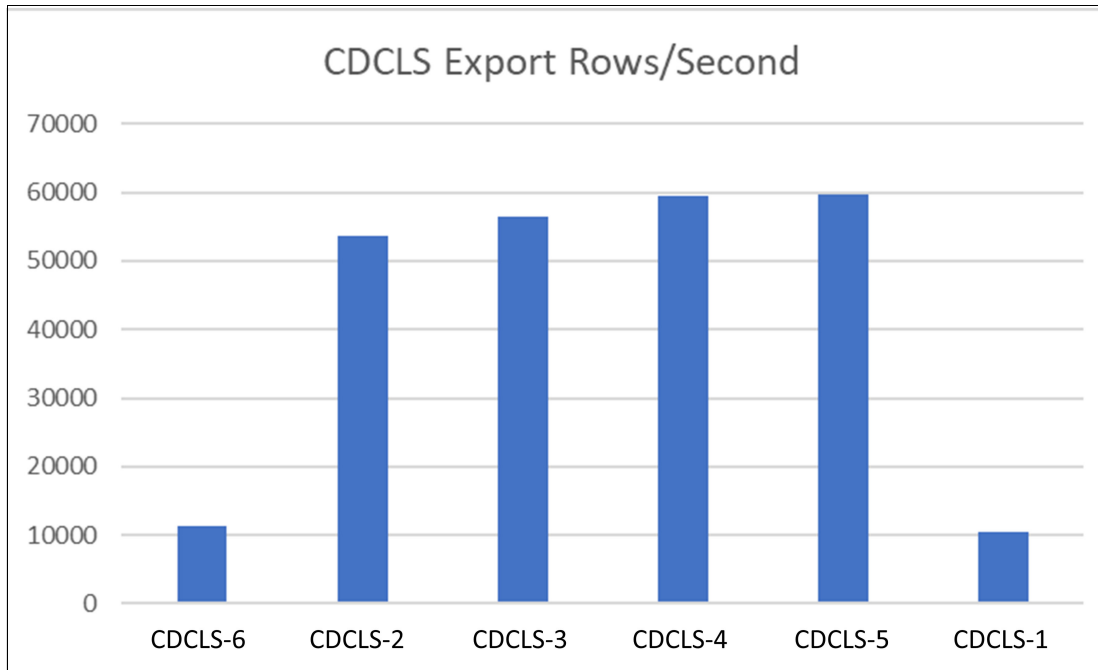


Figure 7-9 CDCLS with performance slowdown in the first and last Split.

You can first determine whether the number of rows exported is equally distributed. If as shown in this case, the first and last split are slower, a non-optimal access plan might be the reason. If this increases the overall runtime, this needs to be fixed.

Because there are different reasons for a non-optimal access plan and different solutions, the best is to open an SAP support incident and optimize this together with SAP support.

One workaround that might help in this situation is to use the Db2 optimizer level “0” for these statements. The optimizer level cannot be set as an R3load option, but you can use the procedure described in *SAP Note 1818503 – DB6: SAP Optimizer Profiles*.

To enable an SAP optimizer profile, you must insert the guideline for optimizer level 0 and the conditions when the guideline should be applied. These are set in the table DB6_OPTPROFILE. This can be done using SAP transaction SE16 or manually as described here.

The table DB6_OPTPROFILE consists of the following columns:

TABNAME	The table name that is referenced in the query,
PATTERN	One or more SQL patterns
INACTIVE	Indicates if the guideline is active. Leave this column empty. To inactivate a guideline, you can use the character “X”.
WP_ONLY	Indicates that this guideline is valid for only SAP work processes. As the guideline should be active for R3load as well, leave this empty.
GUIDELINE	A valid SAP Optimizer Profile Guideline. In our case, a guideline to set the optimization level to 0: <code><QRYOPT VALUE="0"/></code>
REMARK	Some meaningful comment that indicates the reason for the guideline.

The statement in Example 7-5 adds an SAP optimizer guideline into the table DB6_OPTPROFILE that is valid for the table FAGLFLEXA and a SQL statement that performs a SELECT on the table.

Example 7-5 Optimizer Guideline Example

```
INSERT INTO SAPSRC.DB6_OPTPROFILE VALUES ('FAGLFLEXA',
'+[SELECT%FROM%FAGLFLEXA%]', ", ", '<QRYOPT VALUE="0"/>', 'R3load EXPORT');
```

The guideline is applied to all SQL statements that select data from the table FAGLFELXA. So, in this case, all table splits use this guideline. Usually, this is not an issue, but if you want to apply the guideline only for a specific R3load export, you must specify the SQL Statement from that particular *.WHR File.

To verify the guideline, analyze the SQL query with dmctop or another EXPLAIN tool, and validate the optimization level in the output as shown in Example 7-6.

Example 7-6 Optimizer Guideline in EXPLAIN Output

Package Context:

- -----
SQL Type: Dynamic
Optimization Level: 0
Blocking: Block All Cursors
Isolation Level: Uncommitted Read

Furthermore, the SQL statement that is running contains the information about the optimizer guideline. The following statement is an example of an optimizer guideline in a SQL statement:

```
/* optprofile */ /* <OPTGUIDELINES><QRYOPT VALUE="0"/></OPTGUIDELINES> */
```

If the workaround works as expected, you see an index access with appropriate start and stop predicates and the export runs significantly faster. Still, it is advised to open an SAP incident to validate the workaround. For details about dmctop and db2expln, refer to Chapter 10, “Tips for monitoring” on page 175.

7.3.3 Import considerations

Different import options can affect your migration.

Forced LOAD

The sequential Import is used if you start the R3load with the following options:

```
R3load ... -loadprocedure fast LOAD_FORCED
```

This forces only one active R3load process per table. This can be used to achieve an overlapping of export and import.

For the CDCLS such an order-by.txt file might look like Example 7-7.

Example 7-7 OrderBy File Example

```
[CDCLS]
jobNum=1
CDCLS-1
CDCLS-2
```

CDCLS-3
 CDCLS-4
 CDCLS-5
 CDCLS-6
 CDCLS-7
 CDCLS-8
 CDCLS-9

In addition, ensure you set the Db2 database configuration parameter *locktimeout* to *-1* because a parallel R3load process might be scheduled. If you do not set the lock timeout to *-1*, the scheduled but locked R3load fails when reaching the timeout and must be restarted.

Check the output of **db6util -s1** for lock wait situations as shown in the Example 7-8.

Example 7-8 Lock Situation Documented by db6util -s1

```
LOCK WAITS:
-----
          17                21                28
(PID:462) <--      (PID:495) <--      (PID:622)
      R3load                R3load                R3load
                        ^
                        |-----
                        ^
                        |-----
                                (PID:550)
                                R3load
                                22
                                (PID:501)
                                R3load

DETAILED INFORMATION ABOUT LOCKED PROCESSES:
-----
ID      PID      APPL-NAME      HOSTNAME(PART)  MODE RMODE OBJECT TABLE
17      462      R3load         DB-Server(0)
Status  : UOW Waiting
Wkstn   : coe6x002
Last SQL : INSERT INTO "CDCLS" VALUES( ?, ?, ?, ?, ?, ?, ?, ? )
21      495      R3load         DB-Server(0)    Z    Z  TABLE SAPBN7.CDCLS
Status  : Lock Waiting (186 seconds)
Wkstn   : coe6x002
Last SQL : SET CURRENT ISOLATION RESET
28      622      R3load         DB-Server(0)    U    U  ROW
SYSIBM.SYSTABLES
Status  : Lock Waiting (4462 seconds)
Wkstn   : coe6x002
Last SQL : SET CURRENT ISOLATION RESET
```

If you use the **FORCED_LOAD** option and if you want to deviate from the default of creating the indexes after the import is completed, use the environment variable **DB6LOAD_INDEXING_MODE=2** (Incremental) because this significantly improves the index build. The incremental indexing mode can reduce the total runtime by 50%, depending on the size and number of indexes defined for a table.

Note: Ensure that this variable is set only in the environment for the Migration Monitor instance that uses the forced LOAD. Setting this variable for other R3load processes that create the index before the data load slows the processes.

Parallel INSERT

The parallel import with INSERT is used if you start the R3load with the following options:

```
R3load ... -loadprocedure fast
```

This will use SQL INSERT instead of the db2load API. It allows for parallel imports but switches to a slower single thread performance. You also must consider that the R3load process now writes data to the Db2 transactions log files as in addition to importing data and logging performance might become a bottleneck. Most likely, the required IOPS during highly parallel INSERT by R3load is higher than you see in production on the source system or the target system. Monitor the Db2 logging using the MON_GET_TRANSACTION_LOG table function.

Note: Do not use the `-nolog` parameter of R3load when using parallel import with INSERT because this might cause lock wait situations.

In “Performance comparison – Db2 LOAD versus INSERT” on page 108, we showed that the performance advantage of Db2 LOAD over INSERT is sometimes only a factor of 2–5 for cluster tables, INDX-like tables, or tables with large LOB columns. So, these tables are good candidates for table splitting with multiple parallel INSERT operations.

You can combine the parallel INSERT only with the automatic dictionary creation or COMPRESS_ALL. If specified, the options OPT_COMPRESS_ALL and FULL_COMPRESS_ALL are ignored.

Split LOAD

The parallel Import with Db2 LOAD and temporary tables is used if you start the R3load with the following options:

```
R3load ... -loadprocedure fast SPLITTED_LOAD
```

This option combines some of the advantages of Db2 LOAD and INSERT. It parallelizes the load processing without introducing a significant amount of additional logging on the target side. However, it doubles the write workload and increases the read workload as data is written twice and read once within the database.

This concept is applicable for tables that benefit from the db2load API. For example, for the table SWWLOGHIST, we have tested that Db2 LOAD is 15–20 times faster compared to INSERT processing.

For an example calculation, assume a table of 500 GB needs to be imported, and a single LOAD based R3load imports 100 GB per hour and is 20 times faster compared to an R3load using INSERT. If you import with 5 parallel R3load processes into 5 temporary tables, this takes 1 hour. If the internal LOAD FROM CURSOR is 2 times faster than the standard R3load, it takes an additional 2.5 hours for this final copy. In total it takes 3.5 hours, which is equivalent to 143 GB per hour.

With 5 parallel R3load processes with INSERT, it takes 10 hours because those 5 processes achieve only 50 GB per hour. To equal the rate of the use of temporary tables at 143 GB per hour, you need 3 times as many R3load processes with INSERT running, which means 15 R3load processes with INSERT are required.

Although this is a mathematical exercise, not taking all boundary conditions into account, it explains how using the SPLITTED_LOAD might be faster.

The split LOAD option creates temporary tables in the first phase and populates them. As a default these temporary tables reside in the same tablespace as the permanent table. The tables will be deleted after the import finishes and the space is given back to the tablespace and can be reused by other objects.

However, because the multiple temporary tables are large, the database might allocate more space than required for a significant amount of time. If this occurs, you can give the empty pages back to the file system by reducing the tablespace high watermark. This can be done online in the production target system, but the return of the empty pages adds I/O workload to the database server.

To overcome this effect, consider creating a dedicated tablespace for the temporary tables. To do so you have to create a tablespace manually, grant use to the SAP APP Role on this tablespace, and set the environment variable DB6LOAD_TEMP_TBSPACE to reference this tablespace. The temporary tables are then created in the newly created tablespace and after the heterogeneous system copy is completed, the tablespace can be dropped. This is shown in Example 7-9.

Example 7-9 Enabling a dedicated tablespace for temporary split load tables

```
db2 "CREATE TABLESPACE SPLITTEMP MANAGED BY AUTOMATIC STORAGE USING STOGROUP
    IBMSTOGROUP"
db2 "GRANT USE OF TABLESPACE SPLITTEMP TO ROLE SAPAPP"
export DB6LOAD_TEMP_TBSPACE=SPLITTEMP
```

It is essential to closely monitor the Db2 LOAD operations in the Db2 diagnostic log file or while the R3load is running. Monitor the LOAD operations for any decrease of CPU Parallelism and ensure that the importing machine is not overloaded.

You can combine the split load with the different Db2 specific compression options. The options COMPRESS_ALL and FULL_COMPRESS_ALL work with the split load. The OPT_COMPRESS ALL option is also supported but works differently. The compression dictionary is not created when the data is read from the R3load dump files. Instead, the sampling happens when the data is copied from the temporary tables to the final table.

Note: Do not use the deferred table creation option (DEF_CRT) on the R3load when doing a split load. When you use this option you do not improve the import performance and sometimes this causes issues, especially when you are using Db2 11.5.8 or earlier.

Restart of split LOAD

When a failed R3load is restarted, processing is different with any of the split R3load options compared to standard imports. In the standard case, the failed table is either truncated or dropped. This is a fast procedure and typically finishes within a few seconds.

When a R3load process of one of the split fails, data from other already imported splits can be retained and only the failing split needs to be restarted. To make this work, the table is not truncated. Instead, any data imported by the failed split is deleted. The SQL DELETE Statement is similar to the SQL SELECT Statement in the .WHR file.

The deletion of a large portion of data is logged in the Db2 transaction log file. Thus, log full situations can occur when there is not enough space on the log device or when the number of configured log files is completely consumed. See Example 7-10 on page 149.

Example 7-10 Example for an error message with log file full

```
DB21034E The command was processed as an SQL statement because it was not
a valid Command Line Processor command. During SQL processing it returned:
SQL0964C The transaction log for the database is full. SQLSTATE=57011
FUNCTION: DB2 UDB, data protection services, sqlpgResSpace, probe:2860
MESSAGE : ADM1823E
           The active log is full and is held by application handle "<appl_hdl>".
           Terminate this application by COMMIT, ROLLBACK or FORCE APPLICATION.
FUNCTION: DB2 UDB, data protection services, sqlpWriteLR, probe:6680
MESSAGE : ZRC=0x85100009=-2062548983=SQLP_NOSPACE
           "Log File has reached its saturation point"
           DIA8309C Log file was full.
```

To minimize the risk of such a situation, the DELETE processing during restart is performed in a loop and frequently issues a COMMIT statement.

With SPLITTED_LOAD, indexes must be created after the load. The step of creating the primary key index triggers the merge of the temporary tables to the final table. Because no indexes exist, with each loop a table scan is run to identify the data to be deleted. In the worst case, the last split fails and the table scans must read a large table multiple times, which can cause the restart to take a long time. In fact, it might take several days if the table is large.

To overcome this situation, you have the following options:

- ▶ Manually drop the table, clean up the migration logs and repeat the import for the table completely.
- ▶ Create an index that matches the predicates used by R3load. This must be done before the failed R3load process is restarted. After the delete phase is completed, the index must be dropped again manually.

Runtime comparison.

As the examples demonstrate, the best approach depends on the various factors mentioned in this and other chapters.

The results shown in Figure 7-10 on page 150 show the comparison of three import options on two representative tables.

This one simple test has shown that the best option for the table FAGLFLEXA is to use the SPLITTED_LOAD, but for the table COVREF a single LOAD is the most beneficial setup. With a Parallelism of 15, it is likely the parallel INSERT might become the best method. Do not consider the presented test results as a recommendation for any specific approach. The examples are intended to show you that there can be a significant impact on processing time depending on which method is used.

In general, the SPLITTED_LOAD is a good starting option when used with close monitoring. Based on the results seen from your monitoring, you might choose to switch to parallel INSERT or FORCED_LOAD instead.

To minimize the monitoring and test phase, you can use the R3load Option **-discard <NUMBER>**. This option will stop the export after <NUMBER> of rows, so you do not need to perform a full export. This export cannot be used for a final import but can be used to assess the optimal method for the import because only a subset of the data is exported or imported.

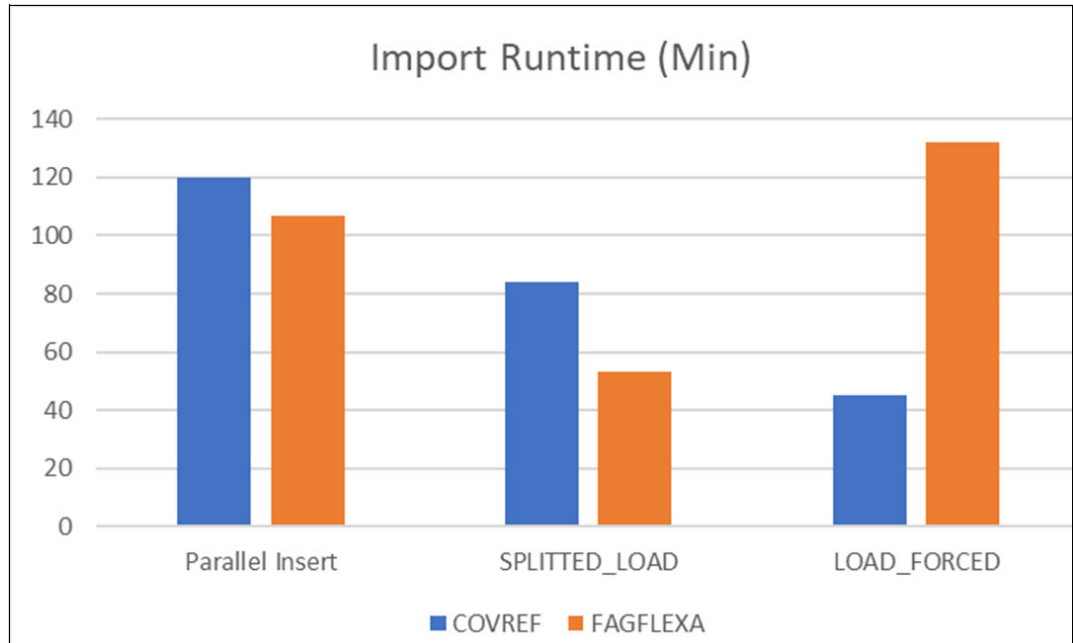


Figure 7-10 Import Runtime Comparison – 5 x Parallel Insert versus 5 x Parallel Load versus single Load

7.3.4 Conclusions

Our general recommendation is that you familiarize yourself with the handling of table splitting. This concept is a powerful option but requires practical knowledge of the tools and Db2. Test this process and the different configurations on a sandbox system.

Recommendations for the export with WHERE splitting

To get the best benefit from WHERE splitting, the data that is being selected should have a high cluster factor based on the index being used. If not, then consider performing a reorganization using the index before starting the export to improve export performance.

The number of exporting R3load processes is also influenced by the number of importing R3load processes and the method chosen.

Recommendations for the import with WHERE splitting

The optimal configuration, number of parallel processes and the import method depends on various factors like infrastructure, table size and definitions.

If you want to create the indexes before the load and if you use the R3load FORCED_LOAD, then we recommend that you enable the incremental index build.

Be aware of the restart conditions.

Typically, multiple large tables are imported concurrently during an SAP heterogeneous system copy. Therefore, you should consider the resource consumption of the different options of table splitting.

Keep the number of splits low. Use 10–20 splits per table or a maximum of 100 GB of data volume per table split to start with.



Data transfer options

The transfer of the data to the cloud requires a fast and secure methodology. The data can be represented by a virtual disk, the export dump files, or database backup images for homogeneous system copies. In this chapter we give an overview of the infrastructure and tools that can be used to migrate your data to different cloud service providers. The section should be used as a starting point to gather information. For details, refer to the cloud service provider of your choice.

The following topics are included in this chapter:

- ▶ 8.1, “IBM Cloud” on page 152
- ▶ 8.2, “Amazon AWS” on page 155
- ▶ 8.3, “Microsoft Azure” on page 156
- ▶ 8.4, “Google Cloud Platform” on page 157
- ▶ 8.5, “IBM Aspera for SAP migrations example” on page 158

8.1 IBM Cloud

IBM offers different options to migrate data to the cloud. These options can be grouped into three main areas:

1. Physical data shipping
2. High-speed network interconnect
3. Data copy tools

Most of these offerings can be accessed by using the IBM Cloud Catalog.

Physical data shipping

The first option involves the shipment of physical devices to move data to the IBM Cloud. These devices can be customer owned or IBM owned devices.

Customer owned devices

Customers who prefer to use devices that they own and control to transfer their data to the IBM Cloud can use the IBM Cloud Data Transfer Service. Within that service, they can send USB devices to an IBM Cloud data center. The transport of the devices must be organized by the customer. The devices are installed in the IBM data center in a rack dedicated to the customer and are connected to the customer network as LUNs, so the customers can remotely control the data transfer from the disks to the Cloud iSCSI destination. After the data is transferred, the disks can be shipped back to the customer location. This offering is ideal when you must transfer large amounts of data and the network bandwidth is limited. The service is free of charge for IBM Cloud customers. Data transfer requests can be initiated using the IBM Cloud Console.

IBM owned devices

Customers who prefer not to use their own devices for the data transfer can engage IBM to organize the data movement on IBM-owned devices. This service is called IBM Cloud Mass Data Migration. The service is based on devices with a capacity of 120 TB that are shipped to the customer location after a request for that service has been started in the IBM Cloud Catalog. Using multiple devices provides a scalable way to transfer large amounts of data if the networking bandwidth is limited. Customers do not need to organize the physical transport, which is part of the service, however this service is fee-based.

Figure 8-1 shows the main use case for physical data shipping by example of Mass Data Migration.

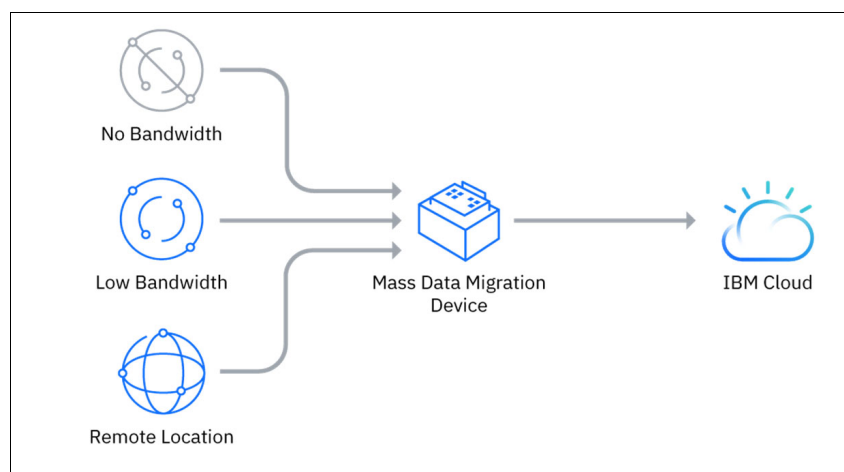


Figure 8-1 Use Case for Mass Data Migration Device – Source IBM website

High-speed network interconnect

In addition to the shipment of physical devices for the movement of data to the IBM cloud, IBM also offers high-speed network connections using IBM Cloud Direct Link Service. When using this service, the customer's on-premises network has a direct connection to the customer's IBM Cloud resources. The customer data is never on the public internet, which is an important security aspect. The customer can choose the required network bandwidth within a range of 50 Mbps to 10 Gbps. That broad range gives the customer the option to migrate data, use the cloud for HA solutions, or store data or backups. To configure and start using the IBM Cloud Direct Link, use the IBM Cloud Catalog.

IBM Direct Link Connect

IBM Direct Link Connect is shown in Figure 8-2 and offers private access to your IBM Cloud infrastructure and to any other clouds linked to your service provider through your local IBM Cloud data center. This option is for creating multi-cloud connectivity in a single environment. With all Direct Link products, you can add global routing that enables private network traffic to all IBM Cloud locations. You can order the service can from the IBM Cloud Catalog.

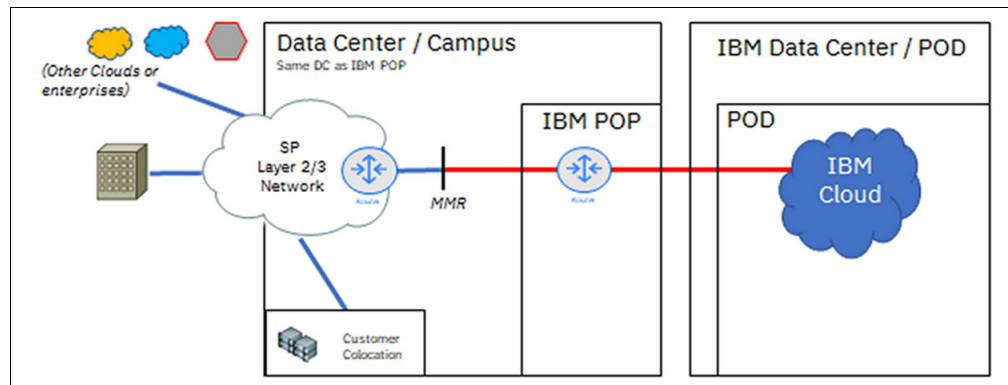


Figure 8-2 IBM Direct Link Connect

IBM Direct Link Dedicated

IBM Direct Link Dedicated allows customers to terminate a single-tenant, fiber-based cross-connect into the IBM Cloud network as shown in Figure 8-3. This offering can be used by customers with colocation premises that are next to IBM Cloud Points of Presence (PoPs) and data centers, along with network service providers that deliver circuits to customer on-premises or other data centers. This service can be ordered by using the IBM Cloud Catalog.

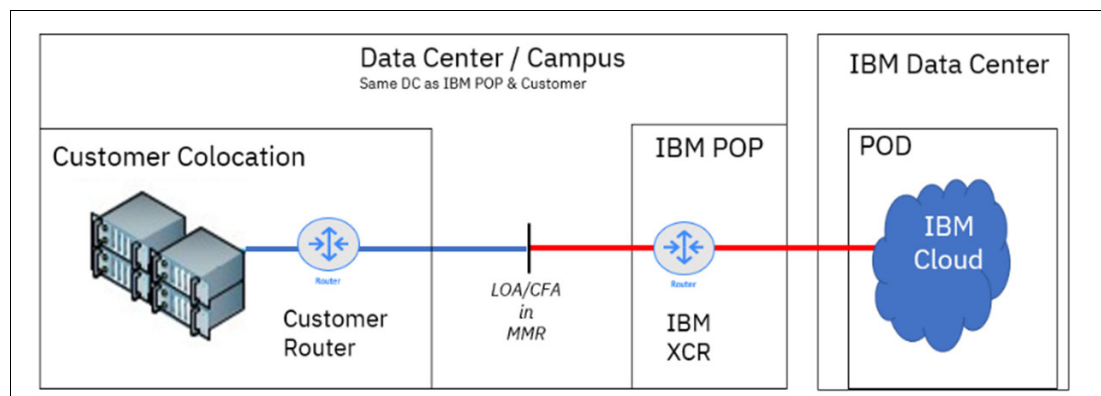


Figure 8-3 IBM Direct Link Dedicated

Data copy tools

IBM offers tools, such as IBM Aspera®, to transfer data to the IBM Cloud. In general, IBM Aspera gives you high-speed data transfer to move large volumes of data to, from, and between on-premises data centers and any major cloud.

The data transfer process is part of every SAP migration project. Optimizing this step can significantly reduce the overall migration downtime. IBM Aspera can be used in several different ways to help optimize migration data transfers. Aspera can be used to enhance data transfer over FTP or HTTP and is also integrated into the IBM Cloud Object Store to enhance data import performance. The SAP migration test environment that we used during the writing of this publication was created using IBM Aspera to optimize the data export from an on-premises installation into a new SAP installation in the cloud.

IBM Cloud Object Store and IBM Aspera

IBM Aspera high-speed data transfer capabilities are integrated into the IBM Cloud Object Storage. This offers an easy-to-use interface to transfer data fast and reliably to and from IBM Cloud Object Storage. The integration makes the IBM Aspera solution suite available for customers to access their storage location in the IBM Cloud. Aspera is known for very fast data transfers compared to traditional TCP data transfer services. Especially for long-distance transfers to locations with limited network bandwidth or networks with a high rate of packet losses. Access to this service is available using the IBM Cloud Catalog.

IBM Aspera high-speed transfer provides advantages for large file transfers when compared to FTP and HTTP approaches. IBM Aspera high-speed transfer uses the IBM FASP® protocol to transfer data. To gain an estimate of data transfer times, customers can use the IBM Asper file-transfer calculator.

Figure 8-4 shows that an IBM Aspera file transfer might be 45 times faster with the given network characteristics for a 100 GB file conveyed by IBM Aspera compared to a traditional TCP-based transfer.

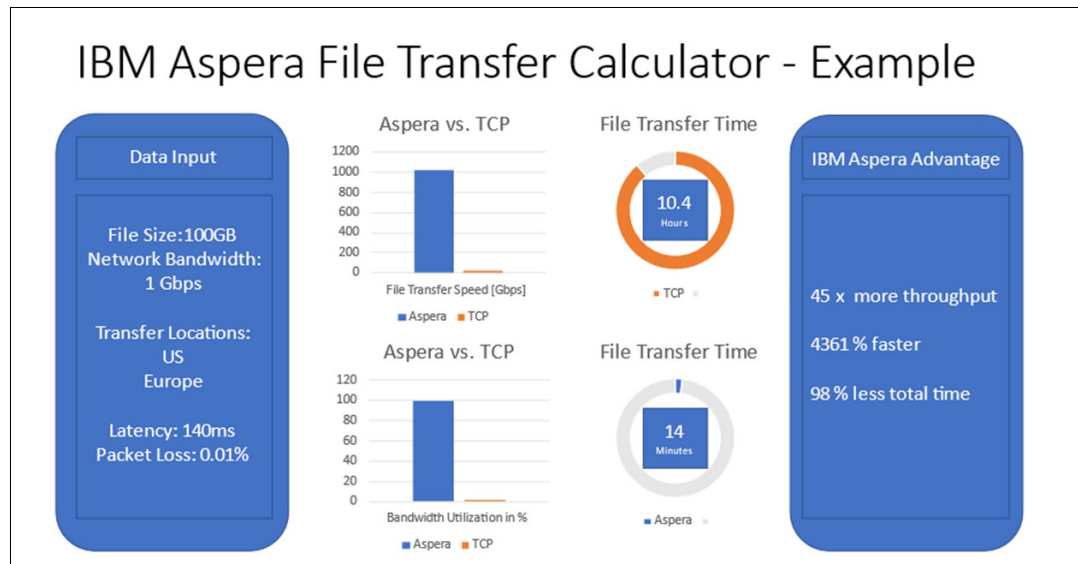


Figure 8-4 IBM Aspera File-Transfer Advantages versus TCP

Section 8.5, "IBM Aspera for SAP migrations example" on page 158 provides more details and a description of a sample configuration for the data transfer with IBM Aspera.

8.2 Amazon AWS

This section provides an overview of the infrastructure and tools that can be used to migrate to Amazon AWS.

Amazon AWS DataSync

Amazon AWS offers DataSync as the way to migrate data to the cloud. This can be deployed in three different ways:

1. The AWS DataSync can be installed into a virtual machine in your data center.
2. It can be installed on an EC2 instance on AWS Outposts, which are gateways to AWS services integrated in the on-premises network by additional hardware.
3. With AWS Snowcone, a DataSync agent comes preinstalled on the device.

High-speed network interconnect

In all three ways, the high-speed network interconnect accelerates the transfer between on-premises and AWS. The data can come from any of the following sources:

- ▶ Network File System (NFS) shares
- ▶ Server Message Block (SMB) shares
- ▶ Amazon Simple Storage Service (Amazon S3) buckets
- ▶ Amazon Elastic File System (Amazon EFS)
- ▶ Amazon FSx for Windows File Server
- ▶ Amazon FSx for Lustre
- ▶ Amazon FSx for OpenZFS
- ▶ Amazon FSx on NetApp ONTAP.

The service includes secure online file transfer by encrypting the traffic.

The increased rate of transfer originates from a special network protocol separate from the storage protocol. This higher performance can be 10 times faster than other ways of transfer.

Amazon AWS Direct Connect

Amazon AWS Direct Connect is a way to connect your on-premises data center directly with the backbone of the AWS cloud. This not only increases the speed, but also as your data never touches the public internet it increases the security of the data transfer. It assures minimal number of hops and minimal latency with maximum speed.

After you set up Direct Connect between your corporate data center and AWS, you can directly transfer files to Amazon Elastic File System (EFS). For this, you can mount Amazon EFS files directly on an on-premises server.

Data copy tools

Alternatively, you can use Storage Gateways. Storage Gateways are virtual appliances that are installed on-premises. They replicate files and block storage. From the source side, they integrate NFS or iSCSI devices

Physical data shipping

For slower network connections it might be an option to use Amazon Snowball. Amazon Snowball is a data transport solution that accelerates moving terabytes or petabytes of data into and out of Amazon Web Services. These appliances use storage devices designed to be secure for physical transport. Thus, they avoid high network costs, long transfer times, and security concerns.

8.3 Microsoft Azure

This section includes an overview of the Microsoft Azure infrastructure and describes the tools that can be used to migrate to a Microsoft Azure cloud instance.

Physical data shipping

Dependent on the network infrastructure between your data center and the target server in Microsoft Azure, the upload of the data can take a lot of time. Instead of uploading the data by using a slow internet connection, it can be beneficial to use Azure Data Box, Data Box Disk, or Data Box Heavy. These appliances differ in size, from 8 TB to 1 PB. These appliances are installed in the local data center. The data is uploaded to the box, and the box is then shipped to Microsoft and uploaded directly to Azure. For the upload in the local data center, NAS protocols such as NFS or SMB/CIFS can be used. The data on the box is encrypted with AES and is highly secured in that way. The box is cleaned after the upload into Azure.

High-speed network interconnect

Appliances like Azure Data Box Gateway or Azure Stack Edge can be used to create a direct connection to Microsoft Azure. Data Box Gateway is a virtual device based on a virtual machine provisioned in your virtualized environment or hypervisor. Alternatively, Azure Stack Edge is a hardware device installed physically into your data center. Both devices reside in your data center and you write data to them using the NFS and SMB protocols. The device then transfers your data to Azure block blob, page blob, or Azure Files.

These devices allow a data transfer by using the mentioned protocols. They support high-performance transfers by directly transferring the data into Azure. The devices have a local cache implemented, dependent on the size of the disks in Azure Stack Edge or the size of the virtual disks in Azure Data Box Gateway. This accelerates the copying of files into the device.

Note: When you are transferring the data for migration purposes, minimize the delay of transferring the data into Azure and configure the cache to allow the immediate transfer of the files.

To establish the connection to Azure, a virtual private network (VPN) can be set up, which is flexible and can be set up immediately. Alternatively, a connection using ExpressRoute can be established. ExpressRoute provides connections directly to the backbone of Microsoft Azure. This is provided by partners of Microsoft. The connection is a dedicated connection with a maximum of 100 Gbps with very low latency, which allows fast uploads.

Both kinds of connections, VPN and ExpressRoute are encrypted so that the security of your data is guaranteed.

Data copy tools

For starting the transfer of the data to the cloud, tools like Robocopy or AzCopy can be used. Both tools encrypt the data that is transferred. In case of interruptions, both tools allow a restart of the upload without having to upload all previously uploaded data again.

8.4 Google Cloud Platform

To create a connection between your data center and the Google Cloud Platform (GCP), there are three options:

1. Usage of a public internet connection
2. Direct peering
3. Google Cloud Interconnect

Public internet connectivity

Public internet connection uses the public internet to create a connection. The throughput can be seen as less predictable in comparison to the other two options. The throughput is limited by the internet service provider's capacity and routing. This solution is advantageous if low costs and high flexibility are a high priority. However, the number of hops within the connection might decrease the performance. Also, you must verify that the security can meet the requirements of your company. This can be mitigated by creating a VPN with GCP, but the endpoints are visible from the internet.

High-speed network interconnect

Direct Peering is available in a limited number of countries, for a limited number of sites. Google guarantees fewer hops. The data transfer does not use public internet. Google requires you to provide an around-the-clock contact with your network operations center.

Google Cloud Interconnect

The third option, connecting with Cloud Interconnect, establishes a direct communication to Google Cloud through Google or one of the Cloud Interconnect service providers. This option guarantees a more consistent throughput for large data transfers. As with direct peering, the Cloud Interconnect does not use public internet. There will be SLAs in place for availability and performance.

Data copy tools

Tools that can be used to upload data from on-premises to the Google Cloud include `gcloud` and `gsutil`.

Gcloud

`Gcloud` is originally implemented to manage your Google Cloud resources and developer workflows. However, one of the available options is a secure copy utility to transfer files to the Google Cloud instance.

Gsutil

`Gsutil` is a Python tool that uses a command line to create and delete buckets, list buckets and objects, move, copy and rename buckets, and other functions. One option is to upload, download, or delete objects. They in turn can be secured by using `https` with TLS. The tool is open source and is continuously enhanced.

Storage Transfer Service

With the help of Google Storage Transfer Service, you can schedule periodic synchronization, with advanced filters based on file creation dates, file names, and the times of day you prefer to import data. Storage Transfer Service copies a file from the data source if the file doesn't exist in the data repository or if the version of the file in the source is different from the one stored in the repository. It can use TLS encryption for HTTPs connections.

It enables you to move or backup data to a cloud storage bucket either from other cloud storage providers or from a local or cloud POSIX file system. This can be set up periodically as part of a data processing pipeline or an analytical workflow.

It does not offer real-time support, meaning that it runs periodically with at least one hour in between two periods. Thus, Storage Transfer Service does not support sub-hourly change detection.

The transfer agents of the Storage Transfer Service require a Docker installation and they run on Linux servers or virtual machines (VMs). To copy data on a CIFS or SMB file system, you can mount the volume on a Linux server or VM and then run the agent from the Linux server or VM.

It is possible to transfer a maximum of 1 billion files that are hundreds of terabytes in size with several tens of Gbps of transfer speed. If you have a larger data set than these limits, we recommend that you split your data across multiple transfer jobs.

Physical data shipping

If you have a limited bandwidth in your internet connection, it might help to use the Google Cloud Transfer Appliance.

This hardware device is installed in your data center. To upload the data, you can use NFS on Linux and UNIX, or SCP and SSH on Windows. The appliance is then sent back to Google where the data is uploaded into the Google Cloud. The data is encrypted on the appliance to assure high security requirements. Dependent on the quality of your internet connection this can still be the fastest solution to upload the data.

8.5 IBM Aspera for SAP migrations example

To minimize the overall downtime for SAP migrations, the data transfer time should be kept as small as possible. To further reduce the downtime, the import of the exported SAP dump files into the new system must begin shortly after the export has been started. This requires that the previously mentioned file transfer must take place in a certain order. In the SAP migration lab we tested how well IBM Aspera fulfills these requirements.

Note: Aspera is available for all major cloud platforms. So, this might become a unified tool if you pursue a multicloud strategy.

8.5.1 Setup description

An overview of the data flow, the server setup, the transferred files, and the directory structure can be found in Figure 8-5 on page 159.

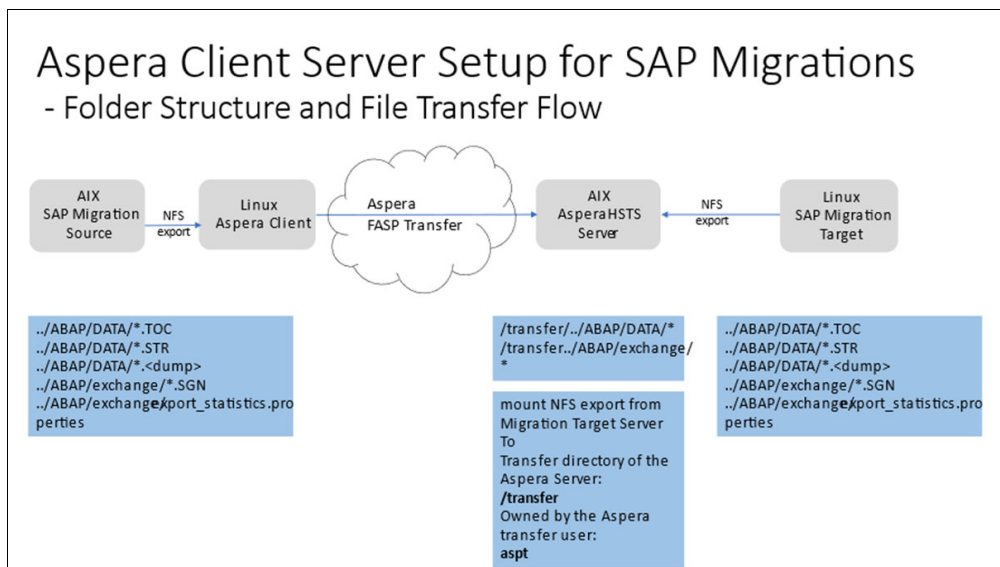


Figure 8-5 SAP Migration Test Lab Aspera Setup

8.5.2 Hardware and software environment

In the SAP migration test lab, we used four distinct AIX LPARS and Linux KVMs to build the SAP migration environment:

1. The SAP source system is installed on an AIX LPAR using AIX 7.2 TL 5 SP 3 and Db2 11.1.4.6.
2. The SAP target system is installed on a Linux KVM using Red Hat 8.5 and Db2 11.5.7.
3. To initiate the file transfer, a Linux KVM system with Red Hat 8.5 and the Aspera client software is used. The Aspera client system has the SAP export directories mounted using NFS. The Aspera client connects using a 1 Gbit connection to the Aspera Server.
4. The Aspera server is using AIX 7.2 TL 5 SP 3 and has IBM Aspera HSTS (High-Speed-Transfer Server) installed. The Aspera server has the import directories mounted for the SAP target system using NFS.

8.5.3 Aspera usage

We used a setup with separate servers for the Aspera client and for the Aspera server to avoid any security issues and any potential runtime impact of running additional software on the SAP production systems. For automation flexibility we decided to use the Aspera command-line client to initiate the data shipping. The Aspera client transfer command was embedded into a shell script, which also controls the order of the file flow. The script itself is started and controlled by using a cron job.

We also tested a setup of two Aspera HSTS servers running the file transfer using watch folders. The watch folder configuration was tested with two options:

1. The SAP files were transferred when they were fully written.
2. We started to transfer the SAP files while they are still open and filled with data.

Both approaches worked well and offered GUI support. In the test, we used the Aspera client set a maximum of limit impact on the SAP system.

8.5.4 SAP files to transfer

The script transfers all files needed for an automatic export and import. First we transfer all *STR files from ../DATA of the Source system to ../DATA of the target system. Then we transfer the ../DATA/<dump files> when the ../exchange/*SGN file signals that the dump files are ready for transfer. The *SGN files are created when an export package is ready for transport and when all associated files especially the dump files are closed. After the last dump file and the associated *TOC is transferred, the *SGN file will be transported. When the *SGN file arrives at the target system, the import of the associated dump files is initiated by the importing Migration Monitor. After all files are imported, we transfer the export_statistics.properties file to finish the migration.

8.5.5 Shell script description

We previously used a shell script for the SAP file transfer from the source to the target system. The script calls the Aspera client, which moves the actual file from source to target.

We started the shell script every 20 minutes as a cron job. To ensure a file is transferred only once, we use flock to set a write lock when the script starts and compiles the list of files to be transferred during the script runtime.

First, the script sends all of the *STR files from the source to the target server. Then it checks whether *SGN files are available and compiles the list of files to be transferred. Based on this list, the Aspera client transfers all files and moves the transferred files to ../DATA/dump_archive. The *SGN files are transferred in a second call of the Aspera client and also moved to ../DATA_dump_archive. As a final step, the script checks to see whether all files are transferred by comparing the number of *STR and *SGN files and transfers the export_statistics.properties file if that condition is fulfilled.

Example 8-1 shows the script that is used for the proof of concept. This example is not intended to be a ready-to-use solution that works in every environment. You might use this as a base and adapt the script to your environment and needs.

Example 8-1 Sample Aspera Transfer Script

```
#!/bin/bash
#
*****
*****
# Transfer SAP migration dump files from Aspera Client on ASPERACLNT to Aspera
Server on ASPERASERVER
# Please set the following environment variable to avoid password prompting or use
public
# key authentication.
# export ASPERA_SCP_PASS=asperatransfer
# delete $EXP_D/STR_COUNTER.txt after each successful SAP migration
#run cron with flock option to avoid ascp overlaps:
#*/20 * * * * flock -xn /tmp/aspera_client_transfer.lock -c 'export
ASPERA_SCP_PASS=asperatransfer ;
/home/aspwatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange/aspera_client
_transfer.sh_5'
#
*****
*****
#Location directories and helper files
EXP_D=/home/aspwatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange
```



```

DUMP_D=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/DATA
SGN_F=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange/signal_
files
SGN2_F=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange/signal
_files_2
#Dump Files

EXP_DATA_F=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange/ex
port_files_data
#Signal Files
EXP_EXCH_F=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/exchange/ex
port_files_exchange
#Location of archived files
SGN_F_A=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/dump_archive/s
ignal_files
SGN2_F_A=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/dump_archive/
signal_files_2

EXP_DATA_F_A=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/dump_arch
ive/export_files_data

EXP_EXCH_F_A=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/dump_arch
ive/export_files_exchange
#Aspera transfer command definitions
MIG_TAR_HOST=ASPERASERVER
MIG_TAR_DATA_LOC=/ASPERACLNT_watch_folder_receive_2/ASPERA_EXPORT/ABAP/DATA
MIG_TAR_EXCH_LOC=/ASPERACLNT_watch_folder_receive_2/ASPERA_EXPORT/ABAP/exchange
MOVE_TRANSFERRED_FILES=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/
dump_archive/
ASCP_LOG_DIR=/home/aswatch/migrate/ASPERACLNT_source/ASPERA_EXPORT/ABAP/dump_arch
ive/aspera_logs
# Create Counter for *STR Files in ../DATA
ls -la $EXP_D/STR_COUNTER.txt &> /dev/null
RC=$?
if [ $RC != 0 ]
then
    ls -la $DUMP_D/*.STR |wc -l > $EXP_D/STR_COUNTER.txt
fi
#Build Aspera Transfer File: EXP_F for the --file-list option
cd $EXP_D
ls *SGN > /dev/null 2>> $EXP_D/err_no_signal_files.out
#Exit Condition if no new SGN Files
RC=$?
if [ $RC != 0 ]
then
date >> $EXP_D/err_no_signal_files.out
exit
else
    ls *SGN > $SGN_F
sed -r 's/.{4}$//' $SGN_F > $SGN2_F
while read file
do
ls "$DUMP_D/$file".* >> $EXP_DATA_F
echo "$EXP_D/$file".SGN >> $EXP_EXCH_F
done < $SGN2_F

```

```

#Aspera transfer DATA Files
ascp -L $ASCP_LOG_DIR --move-after-transfer=$MOVE_TRANSFERED_FILES --mode=send
--user=aspt --host=$MIG_TAR_HOST -k1 -l1G --file-list=$EXP_DATA_F
$MIG_TAR_DATA_LOC
RC=$?
#Check if the transfer was successful and archive the Transfer File      and the
Helper Files
                                if [ $RC != 0 ]
                                then
                                    mv $EXP_DATA_F
"$EXP_DATA_F_A".error_"$(date +%Y%m%d.%H%M%S)"
                                    mv $SGN_F "$SGN_F_A".error_"$(date
+%Y%m%d.%H%M%S)"
                                    mv $SGN2_F "$SGN2_F_A".error_"$(date
+%Y%m%d.%H%M%S)"
                                    exit
                                else
                                    mv $EXP_DATA_F
"$EXP_DATA_F_A".done_"$(date +%Y%m%d.%H%M%S)"
                                fi

#Aspera transfer Exchange Files
ascp -L $ASCP_LOG_DIR --move-after-transfer=$MOVE_TRANSFERED_FILES --mode=send
--user=aspt --host=$MIG_TAR_HOST -k1 -l1G --file-list=$EXP_EXCH_F
$MIG_TAR_EXCH_LOC
RC=$?
#Check if the transfer was successful and archive the Transfer File      and the
Helper Files
                                if [ $RC != 0 ]
                                then
                                    mv $EXP_EXCH_F
"$EXP_EXCH_F_A".error_"$(date +%Y%m%d.%H%M%S)"
                                    mv $SGN_F "$SGN_F_A".error_"$(date
+%Y%m%d.%H%M%S)"
                                    mv $SGN2_F "$SGN2_F_A".error_"$(date
+%Y%m%d.%H%M%S)"
                                    exit
                                else
                                    mv $EXP_EXCH_F
"$EXP_EXCH_F_A".done_"$(date +%Y%m%d.%H%M%S)"
                                    mv $SGN_F "$SGN_F_A".done_"$(date
+%Y%m%d.%H%M%S)"
                                    mv $SGN2_F "$SGN2_F_A".done_"$(date
+%Y%m%d.%H%M%S)"
                                fi
                                fi

STR_COUNTER=$(cat $EXP_D/STR_COUNTER.txt)
SGN_COUNTER=$(ls -la $MOVE_TRANSFERED_FILES/*.SGN |wc -l)
if [ $SGN_COUNTER != $STR_COUNTER ]
then

```

```
        exit
else
    ascp -L $ASCP_LOG_DIR --mode=send --user=aspt -k2 -11G
$EXP_D/export_statistics.properties $MIG_TAR_HOST:$MIG_TAR_EXCH_LOC/
fi
exit 0
```



Special cases

This chapter looks at how to handle some specific special table types that might be present in your SAP environment.

The following topics are covered in this chapter:

- ▶ 9.1, “Special table types” on page 166
- ▶ 9.2, “SAP Business Warehouse based systems” on page 173
- ▶ 9.3, “Db2 pureScale feature” on page 173

9.1 Special table types

Db2 and SAP supports tables with special characteristics, which includes the following table types:

- ▶ Insert time clustering (ITC) tables.
- ▶ Range partitioned tables.
- ▶ Multidimensional clustering (MDC) tables.
- ▶ Column-organized tables (BLU Acceleration or CDE).
- ▶ Hash partitioned tables.

Although range partitioned tables and insert time clustering tables can be used for OLTP workloads, the other table types are designed to support OLAP workloads. For detailed use-cases, refer to *SAP Note 1555903 - DB6: Supported IBM Db2 Database Features*.

We strongly recommend running the SAP report *SMIGR_CHECK_DB6* in preparation for a heterogeneous system copy. This report identifies all tables that derive from the SAP standard definition and can be the base to transfer and adapt the SQL statements with the details of the different table types.

None of these table types can be created by R3load natively. R3load requires the existence of the SQL files that are created by the SAP report *SMIGR_CREATE_DDL*.

Note: Run the reports *SMIGR_CHECK_DB6* and *SMIGR_CREATE_DDL* before all heterogeneous system copies. Furthermore, check the SQL files for the correct syntax. In particular, focus on the tablespace assignment.

For further details, check the SAP Community page about blogs for *SMIGR_CREATE_DDL* and *SMIGR_CHECK_DB6*.

9.1.1 ITC

Insert time clustering tables provide an effective way of maintaining data clustering and easier management of space utilization. In ITC tables, data is clustered by using a virtual column that clusters together rows that are inserted at a similar time.

A heterogeneous system copy provides the opportunity to change the tables type. For ITC tables, the process is supported by the SAP report *DB6CONV* and the output of this report can be used for R3load. In the report, choose to run a conversion on the database level and select **Convert Insert Time Clustering (ITC) Candidates to ICT Tables**. Create the conversion queue but do not run the conversion.

You can export the list of identified tables and use it for further processing with R3load. R3load provides the option *ITC_LIST* and so adds all tables identified by *DB6CONV* to a text file with the name *db6_itc_tablist.txt* and exits the report “*DB6CONV*” without running the queue. R3load appends the syntax *ORGANIZE BY INSERT TIME* to the tables automatically.

If your source system already contains ITC tables, the report *SMIGR_CREATE_DDL* generates the required SQL files and ITC tables are created using the SQL files. To do so, specify the “Keep Source Settings” option for the target database.

Regarding optimization and performance, the findings in this book are accurate, but the appropriate configuration of the Utility Heap and the Load Buffer becomes more important.

9.1.2 MDC

Multidimensional clustering (MDC) provides an elegant method for clustering data in tables along multiple dimensions in a flexible, continuous, and automatic way. This table type can be used with SAP BW only.

You can use the SAP report SMIGR_CREATE_DDL to do either of the following actions:

- ▶ Retain MDC tables that exist in the source system with the Keep Source Settings option
- ▶ Convert suitable tables to MDC tables by using the option Use Multi-Dimensional Clustering (MDC) for BW.

In both cases, SQL files are created and processed by R3load.

Similar to ITC tables, the findings for performance and compression in this book are accurate, but the appropriate configuration of Utility Heap and the Load Buffer becomes more important.

9.1.3 Range partitioned tables

Range partitioned tables create partitions of tables based on static upper and lower boundaries. See Example 9-1 for an example of how to create a range partitioned table.

Example 9-1 Sample statement to create a range partitioned table

```
CREATE TABLE MYTABLE(a INT, b INT) PARTITION BY RANGE(b) (STARTING FROM (1)
EXCLUSIVE ENDING AT (1000) EVERY (100))
```

This form of partitioning is supported by SAP for a limited set of tables. To identify table candidates and to define partitions, use the SAP Partitioning Administrator. For details, refer to *SAP Note 1686102 - DB6: DB6 Partitioning Administrator*.

For those tables, each partition can use its own tablespace for indexes and data. Those can be mutually exclusive, or a subset of the partitions can share the same tablespaces. It is also possible that certain or all partitions can use a global tablespace, which means that the syntax for the CREATE TABLE statement can have many variations. The report SMIGR_CHECK_DB6 checks for inconsistent or wrong assignment of the table and its partitions. The report SMIGR_CREATE_DDL creates SQL files, containing the table DDL and a SQL file for the tablespaces if required.

When SMIGR_CREATE_DDL creates the SQL files, the tablespace definitions for the partitions are transferred 1:1 from the source system to the target system. This means that no standard tablespace names that include the SID of the SAP system should be used. Instead, it is recommended to use tablespace names derived from the table name of the partitioned table with a numeric postfix.

Note: Although many different cases are incorporated in the reports, certain situations might not be handled completely according to the planned layout. Therefore, examine the DDL Statements for range-partitioned tables that are created by SMIGR_CREATE_DDL.

Regarding optimization and performance for export or import of partitioned tables, the findings in this book still apply.

For range-partitioned tables, two advanced options are available:

1. Conversion of non-partitioned tables to range-partitioned tables during the system copy.

You can use the Partitioning Administrator to prepare the conversion for the DB6CONV report or for a manual conversion on a database level. To support the manual conversion, the report also creates SQL files that can be used to create the table with a new partitioning defined.

Instead of using these files on the source system, you can use this during the heterogeneous system copy to create range-partitioned tables on the target system that are non-partitioned on the source system.

2. If you do a split export, the WHR-Files also define distinct ranges to be exported separately.

Instead of using the ranges defined by R3ta or SAPup, you can use the defined ranges for an already range-partitioned table on the source database. This might allow for an improved export by data partition elimination. Data partition elimination refers to the database server's ability to determine, based on query predicates, that only a subset of the data partitions in a table must be accessed to answer a query.

There are various factors that determine whether this improves export performance. Most importantly, sufficient free I/O and CPU resources are required.

Important: Both optimizations require Db2 knowledge and should be validated using a four-eyes principle. A mistake in the manual partition definition can lead to a loss of data.

9.1.4 Column-organized tables – BLU Acceleration

Column-organized tables add columnar store capabilities to Db2 databases, which include data that is stored with column organization and vector processing of column data. Using this table format provides significant improvements to compression and query performance. This type of table is often called BLU Acceleration or CDE tables and is only supported for SAP Business Warehouse and SAP Convergent Charging.

For more information, see [SAP Business Warehouse on IBM Db2 for Linux, UNIX, and Windows 10.5 and Higher: Administration Tasks](#).

In addition, use the information provided in the following SAP Notes:

- ▶ *1819734 - DB6: Use of BLU Acceleration*
- ▶ *1825340 - DB6: Use of BLU Acceleration with SAP BW and Applications Based on SAP BW*
- ▶ *2751102 - DB6: Db2 11.5 Standard Parameter Settings*

Use Db2 11.5.8 or later. There are improvements introduced in the db2load API and thus, import is significantly improved. Figure 9-1 on page 169 shows the improvement of over 40% for a selected table. In addition, the improvement includes reduced CPU consumption, which can be beneficial for when you perform heterogeneous system copies.

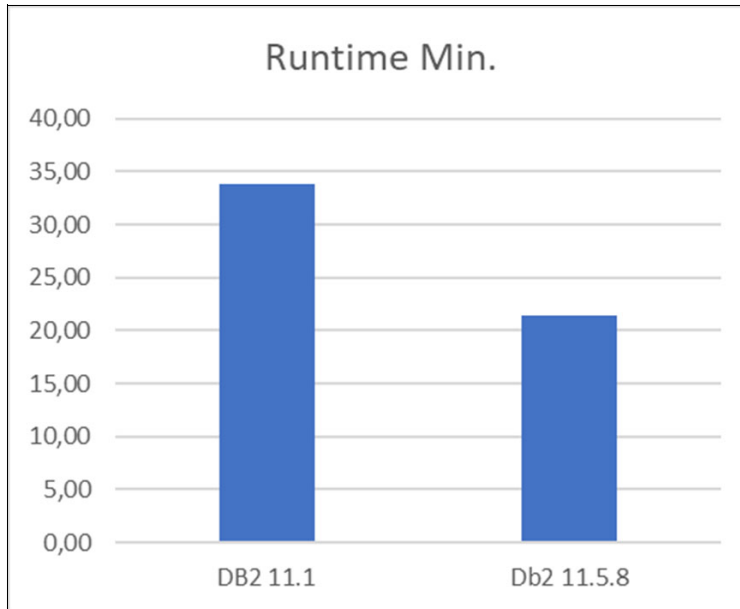


Figure 9-1 Performance Improvement for BLU Load

Column-organized tables have some special characteristics that affect the migration. One of the major differences is the data compression process and how it is enabled. The compression is effective and automated to a large extent. Column-organized tables are always compressed, and there is no option to disable this. The compression is based on a compression dictionary and fully automated. Therefore, the various Db2 compression options of R3load do not affect the import of column-organized tables. That means, the R3load Options COMPRESS, COMPRES_ALL, OPT_COMPRESS and FULL_COMPRESS are ignored by R3load when importing column organized tables.

Compression for column-organized tables is enabled in a way similar to the concept used for default row compression. The compression dictionary is built automatically, and there is no way to affect the procedure by R3load.

With row-organized tables, the compression dictionary is built when the table is populated with 2 MB of data, and while the dictionary is built, no data is written to the tables. The process takes a few seconds. For column-organized tables, the creation of the compression dictionary depends on how the table is populated.

With INSERT and using the command **R3load -loadprocedure fast**, the compression dictionary is built after 500,000 records are inserted independently of the row size. For a table with an average row size of 1 KB, approximately 500 MB of data is sampled.

If the table is populated with the db2load API and the command **R3load -loadprocedure fast LOAD**, 128 GB of data is sampled. If the tables contain less data, the dictionary is smaller of course. Building the compression dictionary might take some time. Therefore, the compression dictionary is created asynchronously, and the data load continues. Data that is not compressed in the first step is automatically compressed asynchronously. The LOAD option is the preferred option as this is by factors faster for the typical BW tables.

The dictionary creation requires memory from the Utility Heap. Therefore, a large Utility Heap is required for loading column-organized tables. Configure at least 1,000,000 pages per table, which is loaded in parallel. In the test environment, we see a maximum of 25% Improvement if you allow more Utility Heap than for row-based tables.

The process of dictionary creation also writes a file to disk that can become 128 GB. Therefore, you must ensure enough disk space and throughput in the directory. For each concurrently loaded table, a maximum of 128 GB space is required.

The files reside in the directory: /db2/<SID>/db2<sid>/NODE0000/SQL00001/load/

In this directory, a subdirectory structure directory is created with one directory for each running LOAD process, and the files are deleted when the LOAD is completed. The files are not deleted after the dictionary is built and remain in the directory until the table is completely loaded. Example 9-2 is an example of the contents of one of these subdirectories.

Example 9-2 Example for temporary LOAD files

```
db2/TAR/db2tar/NODE0000/SQL00001/load/DB20003C.PID/DB200004.OID
[db2tar@db6x141001 DB200004.OID]$ ls -lh
total 100G
rw-r--r-- 1 db2tar dbtaradm 18K Feb  2 22:50 load.CT1
rw-r--r-- 1 db2tar dbtaradm 18K Feb  2 22:50 load.CT2
rw-r----- 1 db2tar dbtaradm  70 Feb  2 22:50 load.loc
rw-r--r-- 1 db2tar dbtaradm 18K Feb  2 22:50 load.min
rw-r--r-- 1 db2tar dbtaradm 680 Feb  2 22:50 load.msg
rw----- 1 db2tar dbtaradm 100G Feb  2 23:14 load.xld.000
rw----- 1 db2tar dbtaradm  39M Feb  2 23:14 load.xld.000.meta
```

With column-organized tables, the LOAD utility remains in the ANALYZE phase of the LOAD process. Therefore, you might see the LOAD remaining in this phase for several minutes as shown in Example 9-3.

Example 9-3 Example for Db2 LOAD phases

```
db6x141001:db2tar 51> db2 list utilities show detail
ID                               = 1
Type                             = LOAD
Database Name                    = TAR
Member Number                   = 0
Description                      = [LOADID: 179.2023-02-02-22.07.34.117536.0 (60;4)]
[*LOCAL.db2tar.230202210757] OFFLINE LOAD CLI          AUTOMATIC INDEXING
INSERT NON-RECOVERABLE SAPSR3.MYBLU
Start Time                      = 02/02/2023 22:07:34.120040
State                            = Executing
Invocation Type                 = User
Progress Monitoring:
  Phase Number                   = 1
    Description                   = SETUP
    Total Work                    = 0 bytes
    Completed Work                = 0 bytes
    Start Time                   = 02/02/2023 22:07:34.120068
  Phase Number [Current]        = 2
    Description                   = ANALYZE
    Total Work                    = 45540091 rows
    Completed Work                = 45540091 rows
    Start Time                   = 02/02/2023 22:07:34.157683
  Phase Number                   = 3
    Description                   = LOAD
    Total Work                    = 0 rows
    Completed Work                = 0 rows
    Start Time                   = Not Started
```

```

Phase Number          = 4
Description           = BUILD
Total Work            = 2 indexes
Completed Work        = 0 indexes
Start Time            = Not Started

```

The Db2 columnar engine is designed for effective parallel processing. Therefore, the CPU resources are efficiently used when loading column-organized tables. However, the CPU usage is significantly greater when loading column-organized tables.

Figure 9-2 shows the CPU usage when just one table is loaded. It consumes 30% of the CPU resources already. So, expect a single digit number of parallel LOADs for column-organized tables.



Figure 9-2 CPU usage of one table load

A summary of this section includes the following information:

- ▶ Use Db2 11.5.8 or higher.
- ▶ All compression options of R3load are ignored.
- ▶ Configure at least 1,000,000 pages for Db2 Utility Heap per concurrent LOAD, but adding more pages is beneficial.
- ▶ Use the R3load Option **-loadprocedure fast LOAD** because it is significantly faster compared to INSERT processing.
- ▶ Plan for a low number of concurrent LOADs.

9.1.5 Hash partitioned tables and database partitioning feature

This type of table is used if the Db2 database is set up with multiple partitions. A partitioned database environment is a database installation that supports the distribution of data across

database partitions. This concept is often referred to as Database Partitioning Feature (DPF) and allows a shared nothing concept for OLAP performance.

Tables in this environment are created to distribute the data evenly across all partitions automatically. The tables are created with a "DISTRIBUTE BY HASH" or as "PARTITIONING ... USING HASHING" clause.

The DDL shown in Example 9-4 is generated for a standard E fact table of an InfoCube.

Example 9-4 DDL file for an E Fact table

```
tab: /BIC/EBFABCUBE5
sql: CREATE TABLE "/BIC/EBFABCUBE5"
("KEY_BFABCUBE5P" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE5T" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE5U" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE51" INTEGER
DEFAULT 0 NOT NULL,
"KEY_BFABCUBE52" INTEGER
DEFAULT 0 NOT NULL,
"/BIC/BCRMEM_QT" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL,
"/BIC/BCRMEM_VA" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BINVCD_VA" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BINVCD_QT" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL,
"/BIC/BRTNSVAL" DECIMAL(000017,000002)
DEFAULT 0 NOT NULL,
"/BIC/BRTNSQTY" DECIMAL(000017,000003)
DEFAULT 0 NOT NULL)
IN "&location&"
INDEX IN "&locationI&"
LONG IN "&locationL&"
PARTITIONING KEY ( "KEY_BFABCUBE51"
, "KEY_BFABCUBE52"
, "KEY_BFABCUBE5T"
, "KEY_BFABCUBE5U"
USING HASHING;
```

This DDL Statement is also created if the system does not have multiple partitions. It is created for all tables that can be distributed across partitions. If you later add one or more partitions, no change in the table definition is required. So, this partitioning type can be ignored during a heterogeneous system copy to Db2.

Note: The recommended process of a heterogeneous system copy to a DPF System is to create a non-partitioned database first, followed by the heterogeneous system copy and an infrastructure change to DPF. The move to DPF requires the data to be redistributed.

9.2 SAP Business Warehouse based systems

Heterogeneous system copies of SAP Business Warehouse (BW) systems or systems based on SAP BW, such as SAP SEM and SAP SCM, require special treatment.

Although they are also exported and imported using the same R3load method as non-BW systems, you must perform special preparation steps on the source system and post-processing steps on the target system because SAP has implemented database-specific features for BW-based systems.

On the source system, you must run the report `SAP_DROP_TMPTABLES` that deletes temporary BW application tables so that they are not unnecessarily copied. Also, on the source system, you must run the report `SMIGR_CREATE_DDL` to retain special table definitions.

On the target system, after database import, you must run the report `RS_BW_POST_MIGRATION` that performs a number of SAP BW specific post migration tasks, which comes with two variants:

1. `SAP&POSTMGR` is used when the DB platform has not changed, for example, when performing a pure Unicode conversion.
2. `SAP&POSTMGRDB` is used when the DB platform has changed. It includes several additional repair operations that are necessary because of the SAP BW implementation differences on the different database platforms.

9.3 Db2 pureScale feature

The Db2 pureScale Feature is a shared disk option for high availability and scalability. Like the Database Partitioning Feature, the process to enable pureScale for the supported SAP application is a heterogeneous system copy to a classical Db2 single node database, after a conversion to Db2 pureScale. Unlike with the DPF feature, a pureScale conversion does not require any data redistribution.

With respect to the heterogeneous system copy, the hints and tips in this book apply because the target is a single node system. However, there are some restrictions regarding the supported table types. The following SAP supported types of tables are not supported in a Db2 pureScale environment:

- ▶ Multidimensional clustering (MDC) tables
- ▶ Insert time clustering (ITC) tables

Therefore, if you plan a subsequent conversion to Db2 pureScale, do not enable these table types and convert tables of this type that currently exist on the source database to regular tables prior to database export. For more details, refer to the *SAP Guide: Running an SAP System on IBM Db2 11.5 with the Db2 pureScale Feature*.



Tips for monitoring

An exhaustive discussion of monitoring database performance is beyond the scope of this book. This section discusses the tools that might be useful.

You can also use other tools such as the IBM Data Management Console, the SAP DBA Cockpit or any other 3rd-party tools.

The same is true for OS monitoring. We determined the tool nmon is useful for ad hoc monitoring and for historical monitoring. Other OS tools are also available:

- ▶ iostat, vmstat, and top on Linux
- ▶ topas on AIX
- ▶ perfmon on Windows.

Also, cloud service providers also provide tools to monitor the system performance. So, you can use IBM Cloud Monitoring, Azure VM insights, AWS Cloudwatch, Google Cloud Monitoring, or 3rd party apps as well.

Use those tools to detect bottlenecks in Db2 or the system environment.

The following topics are discussed in this chapter:

- ▶ 10.1, “Db2 monitoring” on page 176
- ▶ 10.2, “Monitoring a single R3load process” on page 182

10.1 Db2 monitoring

Db2 provides monitoring functions through different interfaces. The interfaces are monitoring table functions, snapshots, stored procedures, or the db2pd utility. They contain the raw data, and, often, a manual calculation of certain values is required.

For example, if you want to get the number of sort operations per interval, you get the total number of sorts by a Db2 monitoring table function. You then must calculate the sort operations per interval by dividing the difference in the total number of sorts by the collection interval.

Db2 also includes tools to automate this processing. Examples are dmctop, dsmtop, db2top, or the MONREPORT modules. Those tools prepare the data but do not provide all the details, compared to the native interfaces.

A different set of tools can provide many details, calculate metrics, and persist data to disk. Examples for these tools include the SAP DBA Cockpit, the IBM Data Management Console, and other commercial tools.

10.1.1 db2top and dsmtop

Db2 includes dsmtop and db2top as non-graphical monitoring tools for a terminal session. The utility db2top was introduced first. Both tools can provide interactive monitoring. They both do auto-refresh of the monitoring data and calculate delta values, which must be done manually when using the Db2 native monitoring functions.

The utility db2top is based on the Db2 snapshot infrastructure that is no longer extended. So, db2top does not benefit from new performance metrics that are added to Db2. The tool is also not available for Windows operating systems.

The tool dsmtop is a Java utility that uses monitoring table functions instead of Db2 snapshots. Because it is written in Java, it is also available for Windows operating systems with some limitations.

An example for monitoring with db2top or dsmtop is to monitor the utilities that are running in the database.

If you select the interactive command **u** of db2top or **Alt-u** in dsmtop, the utilities screen is displayed. If you use the db2load API with R3load, the underlying Db2 LOAD processes are displayed as shown in Example 10-1.

Example 10-1 Example for db2top – Utilities

[-]14:56:12,refresh=2secs(0.022)										
[d=Y,a=N,e=N,p=ALL] Hash # of Utility Utilities										
AIX,part=[1/1],DB2ZWN:ZWN										
Work										
Phase										
Value	entries	Start Time	Type	Pri	State	Type	Work	Unit	rog%	Start Time
32	1	14:48:37.063978	Load	Execute	User		4,893,587	Rows	100%	14:48:37.820626
73	1	14:48:37.822106	Load	Execute	User		1,430,901	Rows	100%	14:48:38.410790
3359535	1	14:48:36.171723	Load	Execute	User		216,746	Rows	100%	14:48:36.242501
4259913	1	14:48:36.606035	Load	Execute	User		4,903,748	Rows	100%	14:48:36.718483
4277119	1	14:48:43.731411	Load	Execute	User		1,924,065	Rows	100%	14:48:44.475456
4608047	1	14:48:44.431682	Load	Execute	User		1,920,457	Rows	100%	14:48:45.444587
4672621	1	14:48:36.180905	Load	Execute	User		623,898	Rows	100%	14:48:36.246057
5260591	1	14:48:36.554057	Load	Execute	User		1,212,508	Rows	100%	14:48:36.640082
5458223	1	14:48:36.545191	Load	Execute	User		3,194,803	Rows	100%	14:48:36.640104
5459052	1	14:48:36.175761	Load	Execute	User		1,394,956	Rows	100%	14:48:36.246437

10.1.2 dmctop

The dmctop utility is a simple text-based tool for monitoring, like dsmtop. The dmctop utility can monitor Db2 version 11.1.0 and later. Beginning with Db2 v11.5.6, the utility is bundled with Db2.

The dmctop utility is a lightweight, low-overhead tool that works in a text-only environment, such as a simple Unix command line. Monitoring is accomplished by using `mon_get_xxx` table functions. Views can be set to fast refresh rates to provide a real-time view of activity in the monitored database.

You can use dmctop to see key performance indicators in the same way that the db2top command works. The dmctop utility is the most recent monitoring utility and should be used instead of dsmtop or db2top. An example is shown in Figure 10-1.

```

db2
--View
--Overview(O)
--Throughput (v)
--Top connections (B)
--Connections (I)
--Statements (S)
--IO (I)
--Locking (L)
--Memory (M)
--Storage (I)
--Other (O)
--Help (H)
--Settings (K)
--Show preferences (P)
--Set visual alert style (S)
--Quit (Q)
[11:15:09 Data mode: delta, Baseline age: 26, Next refresh: 9 secs] DB overview ASX, member=0/11, db2srcsrc
Act session 34
Log used 04

Overview Resource consumption Throughput Contention Timepent
Start date 2023-02-23 CPU usage % 0.76 Transactions/s 1.39 Connections 54 Avg p read time 4.50
Start time 17:32:19 Instance mem committed 218.8M Select stats/s 5.00 Active connections 2 Avg d read time 0.52
Database status ACTIVE Database mem committed 24.90 OS stats/s 1.00 Lock held 5 Sorts/m 39.09
System physical mem 128.90 Bufferpool memory used 7.84 Activities completed/s 0 Lock waits 0 Sort overflows/m 0.00
Shared sort memory used 2.0M Activities aborted/s 0.00 Lock timeouts 0 Hash joins/m 13.18
Storage usage % 58.77 Storage usage % 58.77 Activities queued/s 0.00 Lock escalations/m 0.00 Hash join overflows/m 0.00
Log usage % 0.27 Log usage % 0.27 Head efficiency 0.84 Deadlocks/m 0.00 Hash grp overflows/m 0.00
Log reads/s 4.34 Log reads/s 4.34 Log reads/s 4.34 Threshold violations/m 0.00 Hash grp overflows/m 0.00
Log writes/s 1.90 Log writes/s 1.90 Log writes/s 1.90 Hit ratio 99.97% Avg p write time -
Logical reads/s 30419.27 Logical reads/s 30419.27 Physical reads/s 195.23 Physical reads/s 195.23 Physical reads/s 195.23
Appsc reads/s 434.93 Appsc reads/s 434.93 Appsc reads/s 434.93 Appsc reads/s 434.93 Appsc reads/s 434.93
Writes/s 2.44 Writes/s 2.44 Writes/s 2.44 Writes/s 2.44 Writes/s 2.44
Appsc writes/s 1.57 Appsc writes/s 1.57 Appsc writes/s 1.57 Appsc writes/s 1.57 Appsc writes/s 1.57

Quit: q, Help: h, Metrics: V, Reset baseline: r, Toggle delta value: F,
Freeze display: f, Save preferences: K, Help: h
db2@091001 dmctop 1.0.3.0

```

Figure 10-1 dmctop screenshot

You can use the dmctop utility to collect data in background mode. However, the utility collects only one set of data when it is called in background mode. Furthermore, you cannot combine different areas to monitor. This means, you cannot collect data for the complete database and all connections in one command. To collect the set of data in certain intervals and multiple areas, you can use a script, which is shown in Example 10-2.

Example 10-2 Sample Script for Background Monitoring with dmctop

```

#!/usr/bin/bash
while true
do
/db2/db2src/sql/lib/bin/dmctop -d SRC -b d -f /tmp/dmctop_overview.txt
/db2/db2src/sql/lib/bin/dmctop -d SRC -b l -f /tmp/dmctop_connections.txt
sleep 600
done

```

This script generates two comma-separated files in the `/tmp` directory. One file for the database overview and another file for details about the connections. If the files exist, the data is appended to the files. If the files do not exist, new files are created. The newly created files also contain one header line. This header line exists only once for each file.

You can use the generated files and import the data in a spreadsheet application and analyze the data.

10.1.3 Db2 MONREPORT module

The Db2 MONREPORT module provides a set of procedures for retrieving various monitoring data and generating text reports. One useful procedure is the MONREPORT.DBSUMMARY routine that provides an overview of the current database workload. You can call this routine and specify a collection interval. The routine collects data when started and at the end of the interval, calculates deltas and generates a readable report. This report also contains the time spent metrics and can lead you to the most used resources or the bottleneck of the system. Use the MONREPORT.DBSUMMARY routine with a longer monitoring interval, such as 600–3600 seconds during a period of interest. For example, if you want to monitor during a phase with high I/O workload, you can determine which workload drives that within Db2.

Example 10-3 shows an excerpt of an MONREPORT.DBSUMMARY report on a system with an R3load export from a dedicated remote application server and most of the time, data is shipped over the network.

Example 10-3 Excerpt from MONREPORT.DBSUMMARY report

	%	Total
	---	---

TOTAL_WAIT_TIME	100	14247504
I/O wait time		
POOL_READ_TIME	0	1586
POOL_WRITE_TIME	0	0
DIRECT_READ_TIME	0	393
DIRECT_WRITE_TIME	0	344
LOG_DISK_WAIT_TIME	0	374
LOCK_WAIT_TIME	0	0
AGENT_WAIT_TIME	0	0
Network and FCM		
TCPIP_SEND_WAIT_TIME	99	14230622
TCPIP_RECV_WAIT_TIME	0	96
IPC_SEND_WAIT_TIME	0	319
IPC_RECV_WAIT_TIME	0	19
FCM_SEND_WAIT_TIME	0	10038
FCM_RECV_WAIT_TIME	0	319
WLM_QUEUE_TIME_TOTAL	0	0
CF_WAIT_TIME	0	0
RECLAIM_WAIT_TIME	0	0
SMP_RECLAIM_WAIT_TIME	0	0

10.1.4 Lock wait situations

Run the SAP utility db6util with option s1 to check for lock wait situations. This is shown in Example 10-4 on page 178.

Example 10-4 db6util – Monitor Lock Wait Situations

```

MIGHOST:db2zwn 71> db6util -s1
This is the DB6 (DB2 UDB) utility program for SAP kernel 700.
(I) Option "-s1 1 1" .
(I) Checking for lock wait situations.

```

SNAPSHOT TAKEN AT: 20220513140838

No Deadlocks were detected

LOCK WAITS:

```

-----
                28                66
(PID:1634366) <-- (PID:1749106)
                R3load                db2bp
                  ^                    75
                  |----- (PID:1892590)
                        db6util

```

DETAILED INFORMATION ABOUT LOCKED PROCESSES:

```

-----
ID      PID      APPL-NAME  HOSTNAME(PART)  MODE RMODE OBJECT TABLE
28  1634366      R3load      DB-Server(0)
Status  : UOW Executing
Wkstn   : MIGHOST.wdf.sap.corp
Appl.   : ptype UNKNOWN
Last SQL : INSERT INTO "COEP" VALUES( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? )
66  1749106      db2bp      DB-Server(0)    Z    U    TABLE SAPZWN.COEP
Status  : Lock Waiting (215 seconds)
75  1892590      db6util    DB-Server(0)    Z    IN   TABLE SAPZWN.COEP
Status  : Lock Waiting (16 seconds)
Last SQL : CALL ADMIN_CMD( ( SELECT 'RUNSTATS ON TABLE sapzwn.coep AND
DETAILED INDEXES ALL' FROM SYSIBM.SYSDUMMY1 FETCH FIRST 1
ROWS ONLY ))

```

(I) Connecting to ZWN as user zwnadm.
(I) Successfully connected to database ZWN .
(I) Application snapshot taken at 20220513140838.
(I) Disconnected from database.
db6util successfully finished (exit 0).

10.1.5 Reorganization status

If you use R3load with a COMPRESS option, table reorganization is triggered to build the compression dictionary. You can monitor the reorganization process as shown in Example 10-5.

Example 10-5 db2pd – Monitor Reorganization Status

```
MIGHOST:db2zwn 54> db2pd -db zwn -reorg
```

Database Partition 0 -- Database ZWN -- Active -- Up 0 days 01:36:32

Table Reorg Information:

Address	TbspaceID	TableID	PartID	MasterTbs	MasterTab	TableName	Type	IndexID	TempSpaceID
...	2DA8	6	5	n/a	n/a	RFBLG		Offline 0	5
...	22A8	1	4	n/a	n/a	SOC3		Offline 0	5

Table Reorg Stats:

Address	TableName	Start	End	PhaseStart	MaxPhase	Phase	CurCount	...
Z4AA12DA8	RFBLG	06/03/2022 16:05:58	n/a	06/03/2022 16:05:59	4	Sort	173686	...
04AA122A8	SOC3	06/03/2022 16:09:44	n/a	06/03/2022 16:09:46	4	Sort	100975	...

You can also use the SQL command that is shown in Example 10-6 to monitor the reorganization.

Example 10-6 SQL – Monitor Reorganization Status

```
MIGHOST:db2zwn 47> db2 "SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, SUBSTR(TABSHEMA, 1, 15) AS TAB_SCHEMA,
REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS REORG_TYPE,REORG_STATUS, DBPARTITIONNUM FROM SYSIBMADM.SNAPTAB_REORG ORDER
BY DBPARTITIONNUM"
```

TAB_NAME	TAB_SCHEMA	REORG_PHASE	REORG_TYPE	REORG_STATUS	DBPARTITIONNUM
RFBLG	SAPZWN	BUILD	RECLAIM+OFFLINE+ALLO	STARTED	0

1 record(s) selected.

If you used the COMPRESS option, you can use the following statement to get information about how many rows were compressed as shown in Example 10-7.

Example 10-7 SQL – Monitor Compression Status

```
MIGHOST:db2zwn 83> db2 "SELECT SUBSTR(TABNAME, 1, 15) AS TAB_NAME, REORG_PHASE, SUBSTR(REORG_TYPE, 1, 20) AS
REORG_TYPE, REORG_STATUS, REORG_ROWSCOMPRESSED FROM SYSIBMADM.SNAPTAB_REORG"
```

TAB_NAME	REORG_PHASE	REORG_TYPE	REORG_STATUS	REORG_ROWSCOMPRESSED
RFBLG	BUILD	RECLAIM+OFFLINE+ALLO	STARTED	21844
SOC3	SORT+DICT_SAMPLE	RECLAIM+OFFLINE+ALLO	STARTED	0

2 record(s) selected.

10.1.6 Log space usage

During the import of data or in the index build phase, database logging occurs. In this case, you can evaluate the amount of log space used and the application holding the oldest log entry as shown in Example 10-8.

Example 10-8 SQL – Monitor Log Space Usage

```
MIGHOST:db2zwn 51> db2 "SELECT TOTAL_LOG_AVAILABLE AS LOG_AVAILABLE,TOTAL_LOG_USED as LOG_USED, APPL_ID_OLDEST_XACT
as OLDEST_APPL_ID from SYSIBMADM.SNAPDB"
```

LOG_AVAILABLE	LOG_USED	OLDEST_APPL_ID
3671186068	338637932	22

1 record(s) selected.

```
MIGHOST:db2zwn 58> db2 "SELECT SUBSTR(APPL_NAME,1,10) AS APPL_NAME, APPL_STATUS FROM SYSIBMADM.SNAPAPPL_INFO where
AGENT_ID=22"
```

APPL_NAME	APPL_STATUS
R3load	UOWWAIT

1 record(s) selected.

10.1.7 Sorts

If you use R3load, sorts can occur. For example, during the export or during the index creation phase of an importing process. The command in Example 10-9 on page 181 provides information about active sorts, total sorts, and sort overflows, and therefore, helps to tune Db2 parameters related to sorting.

Example 10-9 SQL – Sorts

```
MIGHOST:db2zwn 70> db2 "SELECT ACTIVE_SORTS, TOTAL_SORTS, SORT_OVERFLOWS from SYSIBMADM.SNAPDB"
ACTIVE_SORTS      TOTAL_SORTS      SORT_OVERFLOWS
-----
                26                41                0
1 record(s) selected.
```

10.1.8 Utility Heap size

Each Db2 LOAD process uses its own data buffer. This buffer is allocated from the Db2 utility heap memory area. If the data buffer is not defined explicitly, Db2 calculates an intelligent default that is based on free space within the utility heap.

If multiple loads are running in parallel, monitor the utility heap to avoid heap full situations. You can do so by either using the Db2 memory tracker tool or the db2top or dsmtop tool.

With active Db2 LOAD processes, memory for load data buffers is allocated from the utility heap. This is shown in Example 10-10.

Example 10-10 db2mtrk – utility heap

```
MIGHOST:db2zwn 9> db2mtrk -d
Tracking Memory on: 2022/06/04 at 14:59:17
Memory for database: ZWN
  utilh      pckcacheh  other      catcacheh  bph (1)    bph (S32K)
  335.3M     384.0K     192.0K     320.0K     315.8M     832.0K
  bph (S16K) bph (S8K)  bph (S4K)  lockh      dbh        apph (14)
  576.0K     448.0K     384.0K     85.8M      21.6M      128.0K
  apph (12)  apph (11)  apph (10)  apph (9)   appshrh
  64.0K      64.0K     64.0K     64.0K     192.0K
```

10.1.9 SQL statement analysis

Analyzing SQL statements can be complex and requires a good understanding of Db2. Db2 provides the EXPLAIN facility to extract detailed information about an SQL query. Along with the native Db2 utilities, you can also use the SAP DBA Cockpit to analyze SQL statements. During a heterogeneous system copy, the SAP system must be offline, so it is difficult to use the SAP DBA Cockpit. You can start a dedicated application server for monitoring purpose only that no business users are allowed to use, or you can use the remote monitoring capabilities of the SAP DBA Cockpit. For details, please refer to the *SAP Guide: Database Administration Using the DBA Cockpit*.

Alternatively, you can use dmctop to extract SQL query information and to generate EXPLAIN information. To do so, start dmctop, press **s** to display the in-flight statements. Look for the Application Name R3load and scroll down to the line by using the arrow key. Press Enter, then press **L** to view the SQL statement that is run. You can export the statement to a file, or you can run the db2exfmt utility for this statement and save the explain output. The explain output is automatically saved in the home directory of the user that calls the dmctop utility.

10.2 Monitoring a single R3load process

This section describes how to monitor a single R3load process.

10.2.1 The List Utilities command

Other than the information available in db2top, dsmtop or the underlying reporting infrastructure, you can use different methods to get more information about the Db2 LOAD process. The LIST UTILITIES SHOW DETAIL command of Db2 displays information about the change to each currently active Db2 LOAD process.

A LOAD process consists of multiple phases. The current phase is marked in the output. In Example 10-11, table BKPF, Phase #2 (LOAD) is the current one and that 3,305,066 rows (out of 17,520,379) have been loaded.

Example 10-11 Db2 "LIST UTILITIES" Command

```
MIGHOST:Db2zwn 34> Db2 list utilities show detail
ID                               = 10
Type                             = LOAD
Database Name                    = ZWN
Partition Number                 = 0
Description                      = OFFLINE LOAD Unknown file type AUTOMATIC INDEXING INSERT
NON-RECOVERABLE SAPZWN .BKPF
Start Time                      = 06/03/2022 14:48:44.431682
State                            = Executing
Invocation Type                  = User
Progress Monitoring:
  Phase Number                   = 1
  Description                     = SETUP
  Total Work                     = 0 bytes
  Completed Work                 = 0 bytes
  Start Time                    = 06/03/2022 14:48:44.431700
  Phase Number [Current]        = 2
  Description                     = LOAD
  Total Work                     = 3305066 rows
  Completed Work                 = 3305066 rows
  Start Time                    = 06/03/2022 14:48:45.444587
  Phase Number                   = 3
  Description                     = BUILD
  Total Work                     = 8 indexes
  Completed Work                 = 0 indexes
  Start Time                    = Not Started
```

10.2.2 Db2 diagnostic log file

The Db2 diagnostic log file, db2diag.log, contains information about each LOAD job. Example 10-12 shows an example for Db2 LOAD Information on the table MARA. It shows that because of memory constraints, the CPU Parallelism was reduced from 11 to 6.

Example 10-12 db2diag.log

```
2023-01-23-15.07.42.734065+060 I26964E616          LEVEL: Warning
PID      : 1594211          TID : 140351612380928 PROC : db2sysc 0
INSTANCE: db2tar          NODE : 000          DB : TAR
APPHDL  : 0-44           APPID: *LOCAL.db2tar.230123135508
AUTHID  : DB2TAR         HOSTNAME: db6x141001
EDUID   : 215           EDUNAME: Db2agent (TAR) 0
FUNCTION: DB2 UDB, database utilities, sqluvtd_route_in, probe:934
DATA #1 : LOADID, PD_TYPE_LOADID, 48 bytes
```

```

LOADID: 215.2023-01-23-15.07.42.734029.0 (-1;-1)
DATA #2 : String, 52 bytes
Starting LOAD operation (S) (3) (1) [db2TAR .MARA].
2023-01-23-15.07.42.944595+060 I27581E555 LEVEL: Warning
PID      : 1594211          TID : 140351612380928 PROC : db2sysc 0
INSTANCE: db2tar          NODE : 000          DB : TAR
APPHDL   : 0-44           APPID: *LOCAL.db2tar.230123135508
UOWID    : 31             ACTID: 1
AUTHID   : DB2TAR         HOSTNAME: db6x141001
EDUID    : 215            EDUNAME: DB2agent (TAR) 0
FUNCTION: DB2 UDB, database utilities, sqluInitLoadParallelism, probe:1097
MESSAGE  : Load parallelism modified from 11 to 6.

```

10.2.3 The List History command

The Db2 command LIST HISTORY can display detailed information about already completed Db2 LOAD operations.

In an SAP environment, the Db2 Registry variable DB2_HISTORY_FILTER=L is temporarily set by SWPM to reduce potential contention on the history file. With this, the Db2 LOAD information is not collected.

If needed, you can unset the registry variable DB2_HISTORY_FILTER to gather the information. This can be useful for an analysis after a test migration because the history contains information about CPU parallelism, disk parallelism, buffer size and the indexing mode in one place.

10.2.4 R3load log file

The log file of a R3load process shows various information. One particularly useful information is reported after a package has been processed. R3load reports the resources used for GENERAL, DATABASE and FILE processing. With this information, you can check which type of workload dominates this package and which actions are promising for further optimization.

For example, if GENERAL times are high and DATABASE times are low during an export, the move to a dedicated remote applications server can be beneficial, but it would not be beneficial if the DATABASE time dominates the process.

Other than the different components, you can assess if there is a resource bottleneck in the System. For example, if the DATABASE real time is significantly higher than the time for usr, the database cannot process data efficiently and is waiting for resources. This potentially points to a resource bottleneck in the system. Example 10-13 shows an example file.

Example 10-13 Example Resource Usage Information in the R3load <package>.log

```

Resource usage: GENERAL   times: 1070.145/394.505/1.854  2.4%/3.5%/1.3% real/usr/sys
Resource usage: DATABASE  times: 17992.269/7611.104/74.732 40.1%/67.3%/54.2% real/usr/sys
Resource usage: FILE      times: 25788.736/3296.825/61.365 57.5%/29.2%/44.5% real/usr/sys

```

Note: The time and resources that are required to compress the R3load dump files are reported as part of the FILE resource usage. Therefore, a high percentage of FILE processing time does not point to an I/O issue for the export dumps files in all cases.



Homogeneous system copy methods

If the Db2 source and Db2 target databases are on the same operating system, you can use several Db2 specific procedures or third-party tools for a system copy. There is a wide variety of tools available. This chapter includes a discussion of some of the features that were used during testing.

The methods in this section also require the typical pre-migration and post-migration steps, such as suspending batch jobs, locking users, disabling interfaces, and setting up RFC communications.

The following topics are discussed in this chapter:

- ▶ 11.1, “Homogeneous system copy with SAP SWPM” on page 186
- ▶ 11.2, “Db2 HADR” on page 186
- ▶ 11.3, “IBM Db2 Shift” on page 189
- ▶ 11.4, “Downtime minimized methods” on page 194

11.1 Homogeneous system copy with SAP SWPM

One of the methods uses a Db2 backup and restore. If you use an online backup and then rollforward the target database after the restore using the transactions log files this can reduce the downtime significantly.

Because this process is documented elsewhere, the details are not provided here. Refer to the chapter “IBM Db2 for Linux, UNIX, and Windows-Specific Procedures” in the [SAP system copy guide](#) for details.

11.2 Db2 HADR

A second option for moving a database to a new target location is to use Db2 HADR to replicate the database and any changes to the new target system running on your cloud infrastructure. Instead of managing the transaction log files manually, Db2 HADR delivers an infrastructure to manage the replication automatically. This section documents some hints to consider whether you were to choose this method.

Generally the use of HADR involves the use of a cluster manager to provide the automated failover. For this use case, the cluster manager is not needed because the failover is done manually during a final migration step.

This method is not intended to provide an HA solution, so it is possible to use different synchronization methods than those used for an HA environment. Choosing the appropriate synchronization method allows you to limit any issues that occur in a fully synchronous methodology introduced by network latency. Evaluate the available sync modes to assess which method fits your environment best – SYNC, NEARSYNC, ASYNC or most likely SUPERASYNC.

If you have an active Db2 HADR cluster with a cluster manager for automated failover, this method can be used by introducing a 3rd auxiliary standby server without automated failover.

The concept is to setup a Db2 HADR standby node in the cloud or remote data center and perform the HADR takeover during the scheduled SAP migration weekend instead of running time-consuming export, transfer, and import activities. See Figure 11-1.

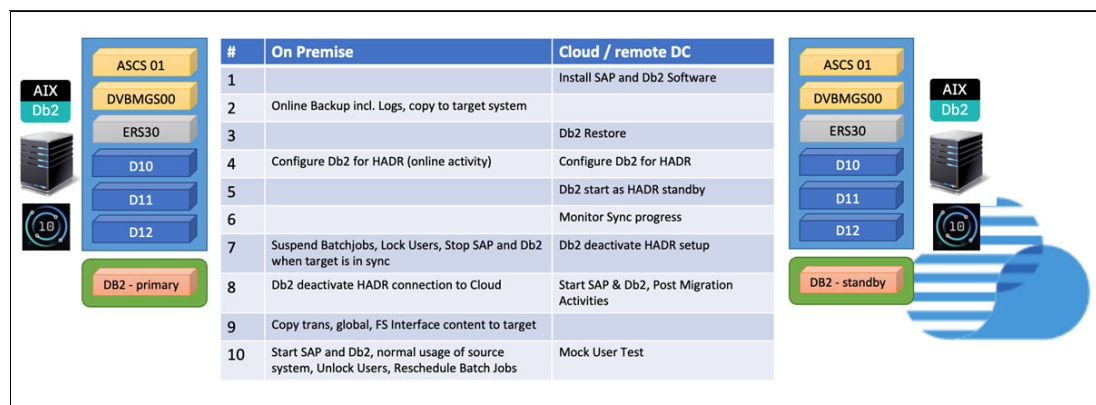


Figure 11-1 Steps for the HADR procedure

11.2.1 Prerequisites and limitations

When configuring an HADR standby database in the new data center it is necessary to meet a couple of requirements, which also define the supported target architecture. This must be considered when choosing the correct provider and available hardware and operating system combination.

The target operating system must be the same version and patch level as the source system. Because rolling OS updates are allowed, a mismatch is acceptable only for a short time, and it is not recommended. The Db2 software version must be the same for the source and target systems.

The details of OS, Db2, network, and storage requirements can be found in the [IBM Documentation for Db2](#).

There are some limitations to using HADR as a system copy method compared to the common SAP SWPM R3load approach. A primary limitation is that you cannot change the database endian type or perform a Unicode conversion.

11.2.2 Procedure

Use the following steps to implement this procedure:

1. The SAP target environment needs to be installed as a plain SAP NetWeaver System that matches your source system version including the Db2 database release. SAP application servers can be prepared with DEFAULT profiles and INSTANCE profiles that match the source environment.
2. Perform a Db2 backup on-premises.
3. Transfer the backup image and Db2 transaction log files to the target.
4. On the target environment, perform a Db2 restore from the backup image.
5. Using the transaction logs, perform a rollforward to the end of logs and keep the database in rollforward mode.
6. Update the source and target databases with the HADR settings.

If the source Db2 is already using HADR on-premises, this can be done as an online activity because the HADR_TARGET_LIST parameter is already configured and must be updated with only the new standby node in the remote data center.

Otherwise, HADR activation can be done in a short system downtime outside of business hours.

Example 11-1 contains the steps to configure the HADR pair if you already have a cluster enabled and with a new auxiliary HADR stand-by for the purpose of moving to the cloud.

Example 11-1 Example commands if HADR is already established on premises

```
### primary on premise
db2 "connect to <SID>"
db2 "update db cfg for <SID> using HADR_TARGET_LIST
<standbyon-prem>:<port>|<newstandbycloud>:<port> immediate"
db2 get db cfg for <SID> | grep HADR
db2pd -db <SID> -hadr

### cloud target
db2stop
db2start
```

```

db2 "update db cfg for <SID> using LOGINDEXBUILD = ON"
db2 "update db cfg for <SID> using HADR_TARGET_LIST
<primaryon-prem>:<port>|<standbyon-prem>:<port> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_HOST <localhostname> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_HOST <primaryon-prem> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_INST db2<SID> immediate"
db2 "update db cfg for <SID> using HADR_SYNCMODE NEARSYNC immediate"
db2 "update db cfg for <SID> using HADR_SPOOL_LIMIT AUTOMATIC immediate"
db2stop
db2start
db2 get db cfg for <SID> | grep HADR
db2 START HADR on DB <SID> as STANDBY
db2pd -db <SID> -hadr

```

Example 11-2 contains the steps to set up the HADR pair of a new HADR stand-by for moving to the cloud only.

Example 11-2 Example of commands if HADR is only used for migration

```

### cloud target
db2stop
db2start
db2 "update db cfg for <SID> using LOGINDEXBUILD = ON"
db2 "update db cfg for <SID> using HADR_TARGET_LIST <primaryon-prem>:<port> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_HOST <localhostname> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_HOST <primaryon-prem> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_INST db2<SID> immediate"
db2 "update db cfg for <SID> using HADR_SYNCMODE NEARSYNC immediate"
db2 "update db cfg for <SID> using HADR_SPOOL_LIMIT AUTOMATIC immediate"
db2stop
db2start
db2 get db cfg for <SID> | grep HADR
db2 START HADR on DB <SID> as STANDBY
db2pd -db <SID> -hadr

### primary on premise
db2 "connect to <SID>"
db2 "update db cfg for <SID> using LOGINDEXBUILD = ON"
db2 "update db cfg for <SID> using HADR_TARGET_LIST <standbycloud>:<port> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_HOST <localhostname> immediate"
db2 "update db cfg for <SID> using HADR_LOCAL_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_HOST <standbycloud> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_SVC <port> immediate"
db2 "update db cfg for <SID> using HADR_REMOTE_INST db2<SID> immediate"
### db2 "update db cfg for <SID> using HADR_SYNCMODE NEARSYNC immediate"
db2 "update db cfg for <SID> using HADR_SPOOL_LIMIT AUTOMATIC immediate"
db2stop
db2start
db2 get db cfg for <SID> | grep HADR
db2 START HADR on DB <SID> as PRIMARY
db2pd -db <SID> -hadr

```

When the HADR cluster is operational, close monitoring is needed to determine if the log gap is decreasing and the HADR members are approaching peer state. Peer state is required before performing the cut over.

During cut over, SAP source application servers must be isolated and stopped like in typical SAP migration scenarios. Then the HADR takeover at the cloud datacenter or the new datacenter can be performed followed by starting the target SAP application servers. The common homogeneous system copy tasks mentioned in the *SAP System Copy Guide*, such as key user acceptance tests and Go-Live would then be run.

This step can either be performed by keeping the HADR cluster working for possible fallback scenarios or by splitting the HADR cluster and removing the original source copy.

This should be decided on a case-by-case basis and depends on the customer requirements.

11.2.3 Customer experiences for an HADR migration

There are customer cases where it was possible to reduce the technical system downtime to less than 10 minutes. This included stopping SAP at the source location, performing the HADR switch and starting SAP in the new cloud environment.

This enables customers to reduce the overall downtime by limiting the system downtime. This assumes that the file system contents of SAP transport directory, global directory, and possible interface directories are synced over before the cut over weekend.

An even more interesting procedure that involves HADR is discussed in 11.2.4, “HADR for heterogeneous system copy” on page 189 which discusses how Db2 HADR can be used for a heterogeneous system copy.

11.2.4 HADR for heterogeneous system copy

In case your future architecture in the remote data center prevents you from using HADR as a migration approach it is possible to implement a temporary platform matching your source architecture and to configure HADR. As an example, customers on IBM Power Systems with AIX, can use this to establish a HADR to Power Systems Virtual Servers on the IBM Cloud in a first step.

During the migration weekend, you can then switch the HADR roles to primary in the remote data center and then perform the SWPM and R3load-based migration approach. The benefit is that the resulting export dump already resides within the new data center and is copied locally.

This approach has been conducted successfully with customers but is an advanced option that requires thorough planning.

11.3 IBM Db2 Shift

You can use system copy tools that are not approved by SAP if the migrations service partner or tool vendor delivers the support for the process. Db2 Shift can be such an example. This can be an option for smaller systems or conversions of many systems.

Note: You cannot change the operating system with this option. So, Db2 Shift can be used for homogeneous system copies only.

The Db2 Click to Containerize family encompasses several tools that provide customers with the ability to quickly modernize their Db2 landscape. The Db2 Shift utility is part of the Click to

Containerize family and can be used to clone a copy of Db2 into a Red Hat OpenShift, Kubernetes, Cloud Pak for Data (CP4D), or an IaaS Db2 instance. The utility is intended to help customers move their existing Db2 databases on Linux into a cloud environment with the minimum amount of effort. Db2 Shift includes the following benefits:

- ▶ Automated, fast, and secure movement of Linux databases to Hybrid Cloud
- ▶ Enables alignment with Agile Delivery and Project Lifecycle with Cloning capabilities

11.3.1 Prerequisites and limitations for the usage of Db2 Shift

The following prerequisites and limitations apply when using Db2 Shift:

- ▶ Passwordless SSH must be configured between source and target system for the Db2 instance user.
- ▶ For an HADR build, the HADR communication ports must be open within the firewall settings.
- ▶ If the online method is chosen, the SAP application servers and the business applications should be stopped, suspended, and isolated.
- ▶ If the offline method is chosen, Db2 Shift must be run once online before using the offline option. This is needed to generate the copy list and generate the move settings for the actual move.
- ▶ For SAP system copies, you must use the `--mirror-path` option. For the `--mirror-path` option, you must use the same `<SID>` in the target as in the source

11.3.2 Features of Db2 Shift

Db2 Shift offers the following features:

- ▶ The ability to move a database without the need to unload, export, decrypt, or backup the database.
- ▶ Shifting of all database settings and objects, which includes external functions that are located in the Db2 library path.
- ▶ Row, columnar, and encrypted databases can be moved.
- ▶ OLTP, SMP, and MPP databases can be moved. At the time of writing, the pureScale installations are an exception and cannot be moved.
- ▶ Easy setup of HADR servers for staged migration.

In addition to directly shifting the database from one location to another, the Db2 Shift program also provides the ability to clone a database for future deployment. This feature is useful for environments where the target server is air-gapped or unavailable for direct connection from the source server.

Finally, the Db2 Shift program has two modes of operation:

- ▶ The Db2 Shift command can be issued with the appropriate options and run directly from a command line or a script.
- ▶ The program can also be run in an interactive mode, with detailed instructions and help for the various shift scenarios.

In summary, by using Db2 Shift provides you can shift your Db2 Linux database into a containerized environment.

11.3.3 Preparing the target and source system

You must first install the target system by using the SAP SWPM before starting the Db2 Shift move. With the current release of Db2 Shift for SAP, it is required to use the same database <SID> as the source system.

Perform the installation with SWPM and Homogeneous System Copy by using the Database Copy Method. Note that passwordless SSH for the db2<sid> instance user is required between the source and target system.

Follow the SWPM installation procedure until the exit step to perform the backup/restore and instead run the Db2 Shift procedure as replacement. Afterwards continue with SWPM to run post migration activities for SAP systems.

Before moving an SAP system, it is required to isolate the system from any updates from applications so as to avoid any kind of communication while starting the new target system. Therefore, suspend batch jobs, lock users, disable interfaces and RFC communications.

11.3.4 Offline move

It is required to build the control files in an online environment. The control files include all items that are moved during the offline processing. The runtime is a few seconds.

1. Generate the control files online and create the target database as shown in Example 11-3.

Example 11-3 db2shift command for creating control files

```
db2shift \  
-mode=move \  
-dest-type=OTHER \  
-ssh \  
-source-dbname=C2C \  
-source-owner=db2c2c \  
-dest-dbname=C2C \  
-dest-owner=db2c2c \  
-dest-server=db2c2c@ibmas-red-t \  
-dest-create-db \  
-mirror-path \  
-online \  
-threads=8 \  
-compress-level=0 \  
-blank-slate=true \  
-verify-only
```

2. Perform the offline move during the downtime while the Db2 instance and all SAP application servers are down and isolated as shown in Example 11-4.

Example 11-4 db2shift command for offline move

```
db2shift \  
--mode=move \  
--dest-type=OTHER \  
--ssh \  
--source-dbname=C2C \  
--source-owner=db2c2c \  
--dest-dbname=C2C \  
--dest-owner=db2c2c \  
--dest-server=db2c2c@ibmas-red-t \  
--dest-server=db2c2c@ibmas-red-t \  
--dest-server=db2c2c@ibmas-red-t \
```

```
--mirror-path \  
--offline \  
--threads=8 \  
--compress-level=0 \  
--blank-slate=false
```

11.3.5 HADR build and move

Perform a move or copy to build a new standby database in the target infrastructure by using the option parameter `--keep-rfw-pending` or `--hadr`, which are equal. A new database is created and is set to the rollforward pending state. Afterward, it is possible to manually recover logs to a specific timestamp or by using the HADR initialization option to enable the HADR Nearsync synchronization method.

1. Copy the database online or offline and enable the rollforward pending mode. This is shown in Example 11-5.

Example 11-5 db2shift command for HADR enabled move

```
db2shift \  
-mode=move \  
-dest-type=OTHER \  
-ssh \  
-source-database=C2C \  
-source-owner=db2c2c \  
-dest-database=C2C \  
-dest-owner=db2c2c \  
-dest-server=db2c2c@ibmas-red-t \  
-dest-create-db \  
-mirror-path \  
-keep-rfw-pending \  
-online \  
-threads=8 \  
-compress-level=3 \  
-blank-slate=true
```

2. Initialization of HADR between source and target using NEARSYNC is shown in Example 11-6.

Example 11-6 db2shift command for HADR initialization

```
db2shift \  
-mode=hadr_setup \  
-dest-type=other \  
-ssh \  
-source-database=c2c \  
-source-hadr-host=ibmas-red-s \  
-source-hadr-port=3700 \  
-dest-server=db2c2c@ibmas-red-t \  
-dest-hadr-host=ibmas-red-t \  
-dest-hadr-port=3700 \  
-source-owner=db2c2c \  
-dest-owner=db2c2c
```

11.3.6 Air-gapped move by using the copy method

The Air-gapped option is used mainly for infrastructures that are not directly connected with each other because of security or other restrictions. Db2 Shift enables the database to be stored into a staging directory, which can be replicated to the new secure disconnected infrastructure. First you must connect to the online database to generate the structure files, which must be copied during the offline/ downtime phase. This is shown in Example 11-7.

Example 11-7 db2shift command for creating control files

```
db2 connect to C2C
db2shift \
-mode=clone \
-source-database=C2C \
-source-owner=db2c2c \
-clone-dir=/tmp/cloneDir \
-online \
-threads=8 \
-blank-slate=true \
-verify
```

After the configuration and structure files are generated, copy the entire database into the staging directory using the Db2 Shift parameter `-clone-dir=/tmp/cloneDir` as shown in Example 11-8.

Example 11-8 db2shift command to copy the database into the staging area

```
db2 terminate; db2 deactivate db C2C
db2stop force
db2shift \
-mode=clone \
-source-database=C2C \
-source-owner=db2c2c \
-clone-dir=/tmp/cloneDir \
-offline \
-threads=8 \
-blank-slate=false
```

If the multi-parallel copy method is successful, copy the entire directory with all its contents into your air-gapped target system before applying the clone. Then, apply the clone by using the `--mirror-path` option, which creates the exact Db2 structure as it was before in your source system, which is required for SAP systems. Optional DBM and DB CFG parameters, such as the `INSTANCE_MEMORY` can be overwritten. See Example 11-9.

Example 11-9 db2shift command to copy the database to the target

```
db2shift \
-mode=Apply_Clone \
-dest-type=OTHER \
-clone-dir=/tmp/cloneDir \
-ssh \
-source-database=C2C \
-source-owner=db2c2c \
-dest-database=C2C \
-dest-owner=db2c2c \
-dest-server=db2c2c@ibmas-red-t \
-dest-create-db \
-mirror-path \
-threads=8 \
```

```
-compress-level=0 \  
-blank-slate=true \  
-overrides="INSTANCE_MEMORY 4024000"
```

11.4 Downtime minimized methods

There are several tools and methods available to further minimize the downtime of a system copy. The tools and processes are designed to reduce the downtime by replicating data from the source to the target database or by creating parts of the database during an online phase.

11.4.1 SAP Database Migration Option in the SAP Update Manager

One option is included in the SAP Update Manager, which is a feature called Database Migration Option (DMO). It incorporates a shadow functionality.

The processing sequence includes the following steps:

1. The SAP Update Manager creates the shadow repository on the database during the online phase. In parallel, the target database is installed and initialized with data such as client.
2. The shadow repository is copied to the target database.
3. During downtime, a classical system copy with R3load is initiated.

After the migration of the application data, which includes data conversion, the update is finalized, and the SAP system runs on the target database.

The process involves an R3load export and import and thus, the findings in this book are valuable for this procedure. For details, refer to the SAP Guide: [Database Migration Option: Target Database IBM Db2 for Linux, Unix, and Windows](#).

The DMO option is not available if the database management system is the same on the source and target system. This means that DMO is not suitable for the use case described in this book, which is migrations with Db2 from on premises to the Cloud. However, it is possible to use the DMO Option to migrate from other database vendors to Db2.

The DMO option is available on request as per *SAP Note: 3207766 - Database Migration Option (DMO) of SUM 1.1 SP01*. The DMO must be requested by reporting an incident on component BC-UPG-TLS-TLA.

11.4.2 IBM Data Replication - Change Data Capture Replication

A different approach that requires less insight in the SAP application is to copy and replicate table data during the uptime of the source system. Typically, the data of only a few tables is replicated with this procedure, and the remainder of the database is copied with the classical R3load procedure.

One example for this procedure is IBM Data Replication - change data capture (CDC) Replication. This tool can be embedded in a heterogeneous system copy with R3load where the largest tables are replicated while the source system is online. This allows you to reduce the downtime significantly and can be used also for heterogeneous system copies with a change of the endianness. It also works for migrations from other selected database vendors to Db2. This is shown in Figure 11-2 on page 195.

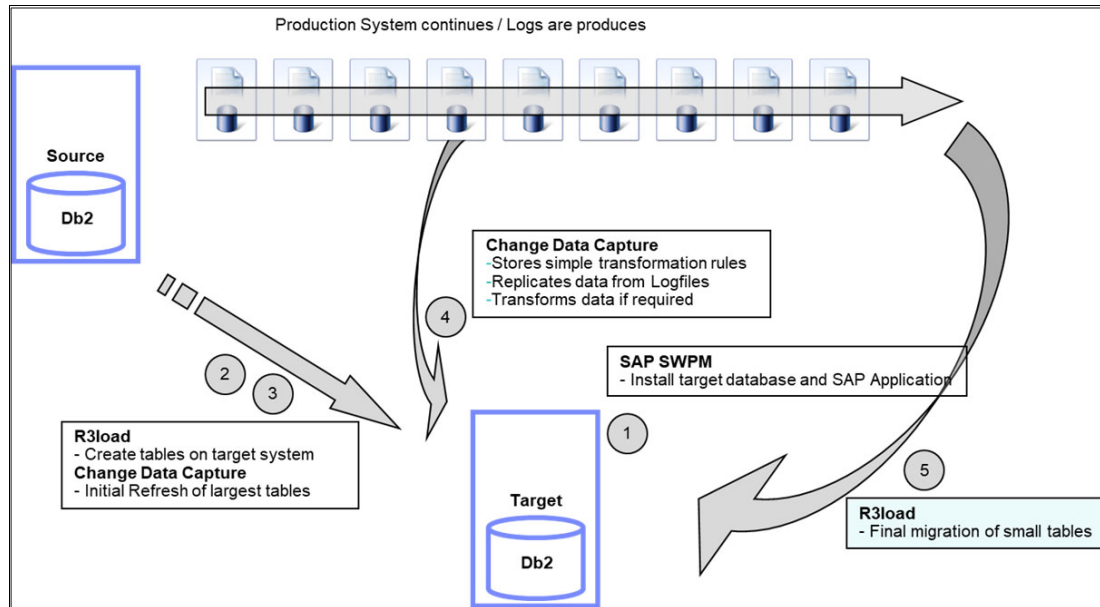


Figure 11-2 Process Overview with IBM Data Replication - CDC Replication

The procedure is divided into 5 steps:

1. Install the target system and Db2 database using the System Copy method of SWPM.
2. Use R3load to create the tables in scope together with their indexes on the target system. R3load is used to ensure that the DDL SQL statements match the SAP requirements.
3. Initiate the initial data transfer with Change Data Capture.
4. Keep data replication running until the source and target tables are in sync and you are ready for the switch.
5. Shutdown the source SAP system and perform a standard heterogeneous system copy and exclude the tables that are within the scope of replication.

Note: This data replication process requires special knowledge and licensing, and a dedicated team is available to run this procedure. You can contact the authors of this book or your IBM representative for details about this procedure.

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 in this list might be available in softcopy only.

- ▶ *Best Practices Guide for Databases on IBM FlashSystem*, REDP-5520
- ▶ *Modernize SAP Workloads on IBM Power Systems*, REDP-5577
- ▶ *SAP HANA on IBM Power Systems Virtual Servers: Hybrid Cloud Solution*, REDP-5693
- ▶ *SAP HANA Data Management and Performance on IBM Power Systems*, REDP-5570
- ▶ *SAP HANA on IBM Power Systems Backup and Recovery Solutions*, REDP-5618

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

Online resources

These websites are also relevant as further information sources:

- ▶ IBM Db2 11.5 for Linux, UNIX and Windows documentation
<https://www.ibm.com/docs/en/db2/11.5>
- ▶ System Copy for SAP ABAP Systems Based on UNIX: SAP HANA 2.0 Database - Using Software Provisioning Manager 2.0
https://help.sap.com/docs/SLT00LSET/afec789b34ba4a5fb5a26826fc8a2584/276253812bec452c8a63e61831db791c.html?version=CURRENT_VERSION_SWPM20
- ▶ FAQ of IBM Cloud for SAP
<https://cloud.ibm.com/docs/sap?topic=sap-faq-ibm-cloud-for-sap>
- ▶ SAP on IBM Power Systems Virtual Server
<https://wiki.scn.sap.com/wiki/display/VIRTUALIZATION/SAP+on+IBM+Power+Systems+Virtual+Server>
- ▶ SAP Sizing
<https://www.sap.com/about/benchmark/sizing.html>
- ▶ Troubleshoot and monitor Linux system performance with nmon
<https://www.redhat.com/sysadmin/monitor-linux-performance-nmon>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



SG24-8531-00

ISBN 0738419044

Printed in U.S.A.

Get connected



