IBM

# PeopleSoft V8 on zSeries Using Sysplex Data Sharing and Enterprise Storage Systems

- zSeries sysplex data sharing and PeopleSoft
- DB2 Connect and load balancing
- ESS performance with PeopleSoft

Viviane Anavi-Chaput
Troy Callanan
Fred Orosco
Dave Perrault
Dipendra Waykar
Bill Worthington



Redbooks

ibm.com/redbooks

International Technical Support Organization

# PeopleSoft V8 on zSeries Using Sysplex Data Sharing and Enterprise Storage Systems

March 2004



SG24-6093-00

**Note:** Before using this information and the product it supports, read the information in "Notices" on page vii.



**First Edition (March 2004)**

This edition applies to PeopleSoft Version 8.4, DB2 for z/OS Version 7, and DB2 Connect EE Version 8.

# Contents

# Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates 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. 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 IBM's application programming interfaces.

# Trademarks

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

AIX®
DB2®
DB2 Connect™
DRDA®
Enterprise Storage Server®
ESCON®
FICON™
FlashCopy®
Hiperspace™
IBM®
Informix®
MVS™
OS/390®
OS/400®
Parallel Sysplex®
PR/SM™
Redbooks (logo) ™
RMF™
S/390®
Seascape®
TotalStorage®
WebSphere®
z/Architecture™
z/OS®
zSeries®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, 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 Sun Microsystems, Inc. in the United States, other countries, or both.

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

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

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

# Preface

This IBM® Redbook describes the implementation of PeopleSoft 8.4 on zSeries® Parallel Sysplex® data sharing environment. The book discusses the benefits of running PeopleSoft in a sysplex with data sharing. It explains the performance considerations and issues when implementing PeopleSoft in a data sharing environment.

The book also investigates the load balancing of PeopleSoft workloads using DB2® Connect to connect to the sysplex DB2 data sharing members. It describes functional and volume tests of load balancing. A section about Enterprise Storage Systems describes the PeopleSoft performance benefits when using ESS.

## The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Viviane Anavi-Chaput** is a Senior IT Specialist for DB2, BI, and ERP at the IBM International Technical Support Organization, New York. She writes extensively, teaches worldwide, and presents at international conferences on all areas of Business Intelligence, ERP, and DB2 for OS/390®. Before joining the ITSO in 1999, Viviane was a Senior Data Management Consultant at IBM Europe, France. She was also an ITSO Specialist for DB2 at the San Jose Center from 1990 to 1994.

**Troy Callanan** is a DB2 Database Administrator at 3M. He has worked with DB2 since 1992 and with PeopleSoft since 1998. Troy holds a B.S. degree in Business Computer Information Systems from St. Cloud State University.

**Fred Orosco** is an IT consultant at the IBM Silicon Valley Laboratory - Integration Center for DB2, specializing in PeopleSoft and Siebel integrations on zSeries solutions.

**Dave Perrault** is an I/T Specialist in the IBM Advanced Technical Support organization, providing technical support to PeopleSoft customers with implementations on DB2 for the IBM zSeries platform. Dave has more than 15 years of experience in large systems and has been with IBM since 1996. With IBM, he has provided technical support for Business Partners porting

applications to IBM DB2 database product offerings, including porting assessments and feasibility studies, benchmarking/performance analysis, and pre/post-sales customer technical support.

**Dipendra Waykar** is an Advisory Software Engineer in PeopleSoft Integration Center at IBM, Silicon Valley Lab. He has more than 11 years of experience in database-server and client-server technologies. Before joining PeopleSoft Integration Center in late 2002, Dipendra was a lead engineer at IBM Informix® Data Management Division and has extensively worked on development and testing of database servers and client APIs.

**Bill Worthington** is an IBM Consulting IT Specialist in the United States. He has 43 years of experience in data processing, including 39 years in technical sales support within IBM. His areas of expertise include disk storage products. In his current position, Bill is supporting storage products used by independent software vendors including PeopleSoft.

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

Cleve Bench, John Grey, Randall Hicks, Mark Hoernemann, Jon Nolting, PeopleSoft Inc., Pleasanton, CA, US

Tom Kennelly, Michael Curtis
IBM ERP Competency Center, US

## Become a published author

Join us for a two- to six-week residency program. Help write an IBM Redbook dealing with specific products or solutions while getting hands-on experience with leading-edge technologies. You will team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

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 Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  **ibm.com**/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

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

1

# Benefits of zSeries sysplex data sharing for PeopleSoft

This chapter introduces the sysplex data sharing benefits for PeopleSoft. We discuss the following topics:

- Why Parallel Sysplex and data sharing for PeopleSoft?
  - Continuous availability
  - Processor scalability
- Parallel Sysplex architecture
- Data sharing architectures
  - Data sharing configurations for high availability and scalability

## 1.1 Why Parallel Sysplex and data sharing for PeopleSoft

There are several motivations for pursuing a PeopleSoft implementation based on zSeries Parallel Sysplex and DB2 data sharing. Many customers are deploying PeopleSoft applications in support of business-critical operations. Typical business drivers include a desire to run a single global PeopleSoft instance, customer and supplier access to Web-based PeopleSoft applications around the clock, and support of 24x7 manufacturing and distribution operations. Those business drivers lead to the following IT requirements:

- Near-continuous system availability
- Central processor scalability through horizontal growth

### 1.1.1 Continuous availability

These business applications require what is known as *near-continuous availability*. Availability in this context means the ability of all or some of the end users and applications to access the production databases. The question of what percentage of work can be processed on a continuous basis is an economic decision based on cost and the value of continuous availability to the organization. Although there is no standard definition of near-continuous availability, many large companies have adopted a goal of no more than five to ten hours per year of planned plus unplanned outages. This translates to an overall system availability of 99.9%. This concept of availability contrasts to the historical paradigm of a maintenance window of 8 hours every weekend, or 400 hours of outage per year. Other companies are adopting service level objectives slightly less demanding but still challenging. For example, an objective of a quarterly four-hour maintenance window (that is, 16 hours per year of planned outage) requires many of the same design principles as the near-continuous availability objective.

Continuous availability is a combination of high availability and continuous operations.

#### High availability

The first component of continuous availability is high availability. High availability is the ability to avoid unplanned outages. This is typically achieved through reliable quality components, internal redundancy, and sophisticated recovery or correction mechanisms. This applies to software as well as hardware. z/OS has millions of lines of code devoted to correction and recovery. This reliability coupled with a design that eliminates single points of failure at the component level provides what is known in the industry as *high availability*. In fact, this is *high reliability*, the ability to avoid unplanned incidents. To achieve continuous

availability we strive for a design that has 99.999% reliability, which equates to less than five minutes of unplanned downtime per year.

#### Continuous operations

The other component of continuous availability is continuous operations. Continuous operations is the ability to avoid planned outages. This includes the ability to upgrade and maintain the central processors, the operating systems, DB2, and coupling facilities without taking the PeopleSoft database down. It also means that DB2 database maintenance must be done while applications are reading and writing data. This requires Online Reorganization, Online Image Copy, and the ability to non-disruptively *flash* or *snap* volumes of data.

### 1.1.2 Processor scalability

Historically systems grew *vertically* by adding processors to the machine (or CEC) or by introducing faster processors. This approach limited the size of a system to the largest single CEC. Systems were also constrained by the amount of data and control information that could be held in the primary DB2 address space DBM1. zSeries Parallel Sysplex architecture enables us to overcome these constraints by clustering multiple CECs in a single operating system image including DB2 data sharing. This enables scalability through horizontal growth of both processor power (MIPs) and virtual storage. Parallel Sysplex also gives us workload management capabilities, as we can now level multiple workloads across two or more machines.

## 1.2 Parallel Sysplex architecture

A fundamental capability to support both high availability and processor scalability is the technology of *clustered* database servers operating against a single copy of the data.

zSeries Parallel Sysplex technology uses a form of clustering where all servers appear to the business application as a single server. This form of clustering, known as *single system image*, enables Parallel Sysplex to provide industry-leading availability by enabling workloads to be balanced across multiple servers. Up to 32 servers can be linked together with near-linear scalability, building on and extending the strengths of the zSeries hardware architecture. Every server in a zSeries Parallel Sysplex cluster has access to all data resources, and every cloned application can run on every server. Using zSeries coupling technology, Parallel Sysplex technology provides a *shared data* clustering technique that permits multi-system data sharing with high-performance read/write integrity. This shared data (as opposed to “shared

nothing") approach enables workloads to be dynamically balanced across all servers in the Parallel Sysplex cluster.

PeopleSoft with its database on DB2 for z/OS® can fully utilize zSeries Parallel Sysplex, which enables PeopleSoft to take advantage of the aggregate capacity of multiple servers for what is probably the most critical part of a PeopleSoft installation, the database server. zSeries Parallel Sysplex helps to ensure maximum system throughput and performance during peak processing periods. In the event of a hardware or software outage, either planned or unplanned, workloads can be dynamically redirected to available servers, providing continuous application availability.

Figure 1-1 introduces a high-level picture of the elements of a zSeries Parallel Sysplex shared data architecture.

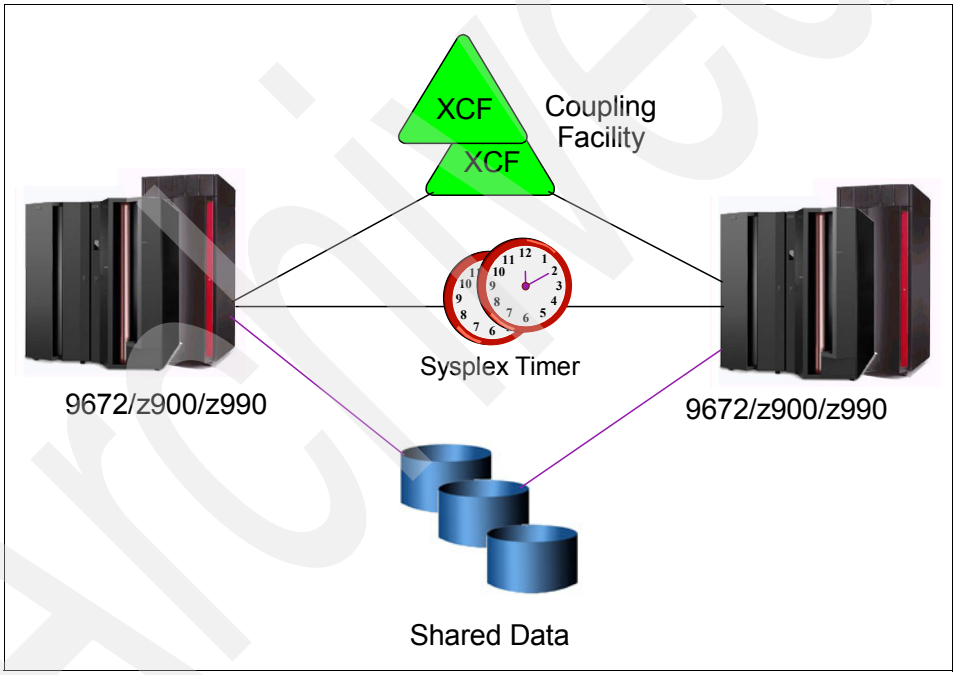



*Figure 1-1 zSeries Parallel Sysplex shared data architecture elements*

In Figure 1-1 we see that a Parallel Sysplex system infrastructure is typically made up of two or more computer systems known as a Central Electronic Complex (CEC), two or more special purpose zSeries computers known as Coupling Facilities (either internal, ICF, or external) for the storing of shared Operating System and DB2 structures between the CECs, two external time sources called sysplex timers, sysplex-wide shared data, and multiple

high-speed, duplexed links connecting the components. This implementation employs hardware, software, and microcode.

As zSeries Parallel Sysplex system is designed for continuous operation, you should make sure that your configuration follows the principle of avoiding single points of failures (SPOFs) to really achieve your availability goal. The key design point is that the configuration should tolerate a failure in any single major component. The design concept of redundancy also applies to zSeries Parallel Sysplex. If at least two instances of a resource exist, then a failure of one allows the application to continue. Sysplex timer, coupling facility, and coupling facility links are elements that should be duplicated.

## 1.3 DB2 data sharing architectures

In Figure 1-2 we complete the picture by laying multiple DB2 data sharing members on top of the zSeries Parallel Sysplex infrastructure.

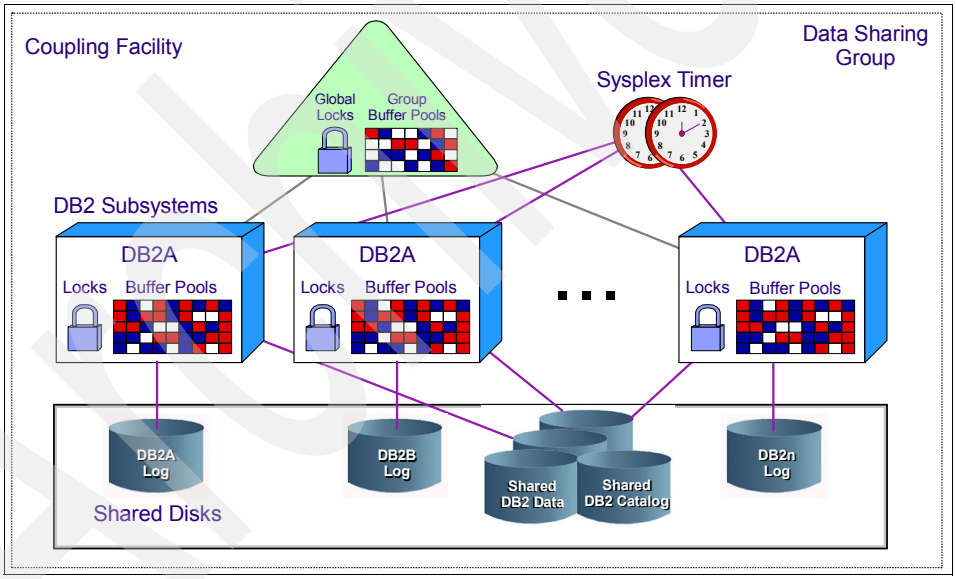



*Figure 1-2 DB2 data sharing in a Parallel Sysplex*

In this picture we see up to 32 DB2 subsystems (also called DB2 members) making up a DB2 data sharing group. There will be one DB2 data sharing group for each production PeopleSoft system. Each DB2 subsystem has its own set of DB2 logs, local buffer pools, and local locks managed by a companion IRLM. The DB2 data sharing group shares the DB2 databases, the DB2 catalog/directory,

and the DB2 data sharing structures, such as the SCA, global locks, and group buffer pools, stored in the coupling facility.

### 1.3.1 Data sharing configurations for high availability and scalability

We complete the DB2 data sharing infrastructure picture with data sharing configuration options for PeopleSoft.

There are four basic data sharing options for providing a highly available environment for a PeopleSoft database using DB2 on z/OS:

- Single active DB2 member with passive (inactive) standby member
- Two active DB2 members without passive standby members
- Two active DB2 members each with a passive standby member in opposite LPAR
- Two active DB2 members each with a passive standby member in independent LPAR

#### *Single active DB2 member with passive standby member - option 0*

The objective of this configuration is availability. Figure 1-3 on page 7 introduces the notion of *primary* DB2 members, which normally have PeopleSoft application servers attached doing productive work, and *standby* DB2 members, which are normally in hot standby mode with no attached applications servers.

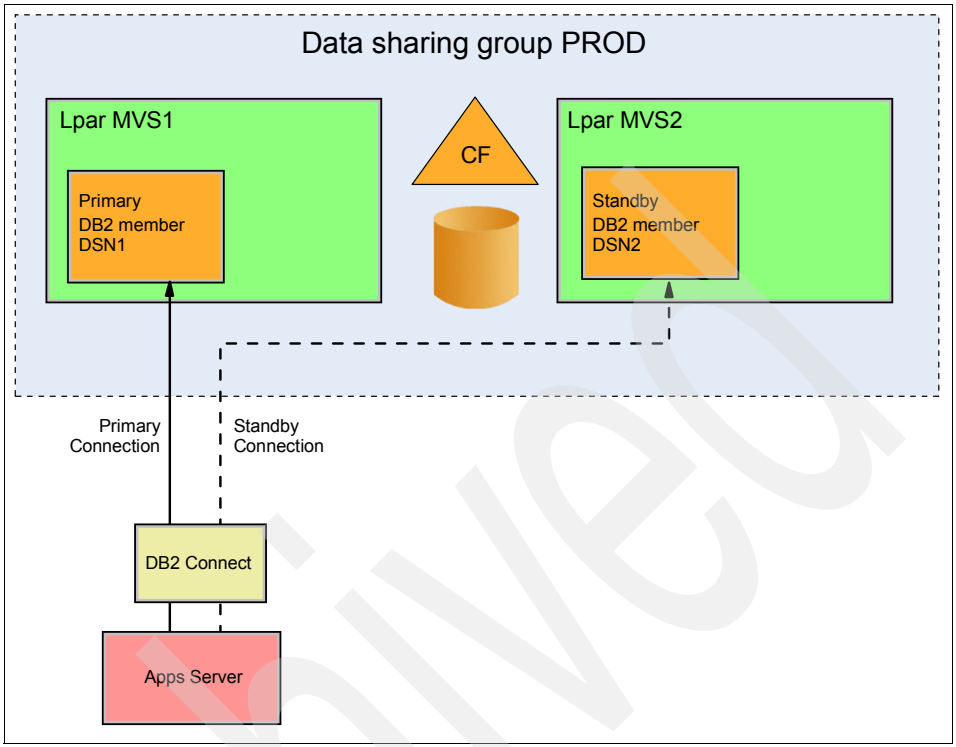



*Figure 1-3 Single active DB2 member with passive standby member*

Each PeopleSoft application server goes through DB2 Connect™ to connect to a DB2 member on z/OS. In the event of a planned or unplanned incident which makes a DB2 member unavailable, DB2 Connect recognizes the need to failover, looks for standby information in its setup definitions, and connects the PeopleSoft application server to the standby DB2 member.

This option is chosen most often when high availability is the main concern and the current hardware is sufficient to handle all of the PeopleSoft database requirements. In other words, your PeopleSoft database workload can be handled by one DB2 data sharing member in one CEC.

Note that the LPAR MVS2 may have other non-PeopleSoft work assigned, but the load there should be light enough that the occurrence of PeopleSoft failover will not cause it to suffer degraded performance.

#### *Two active DB2 members without passive standby members - option 1*

The objective of this configuration is processor scalability and virtual storage relief; it is not availability. The DB2 member DSN2 performs work on behalf of PeopleSoft in normal operation; it provides access to the additional processing

power found in MVS2, and it provides an additional address space of 2 GB of storage for DB2 buffer pools, locks, and prepared SQL (Figure 1-4).

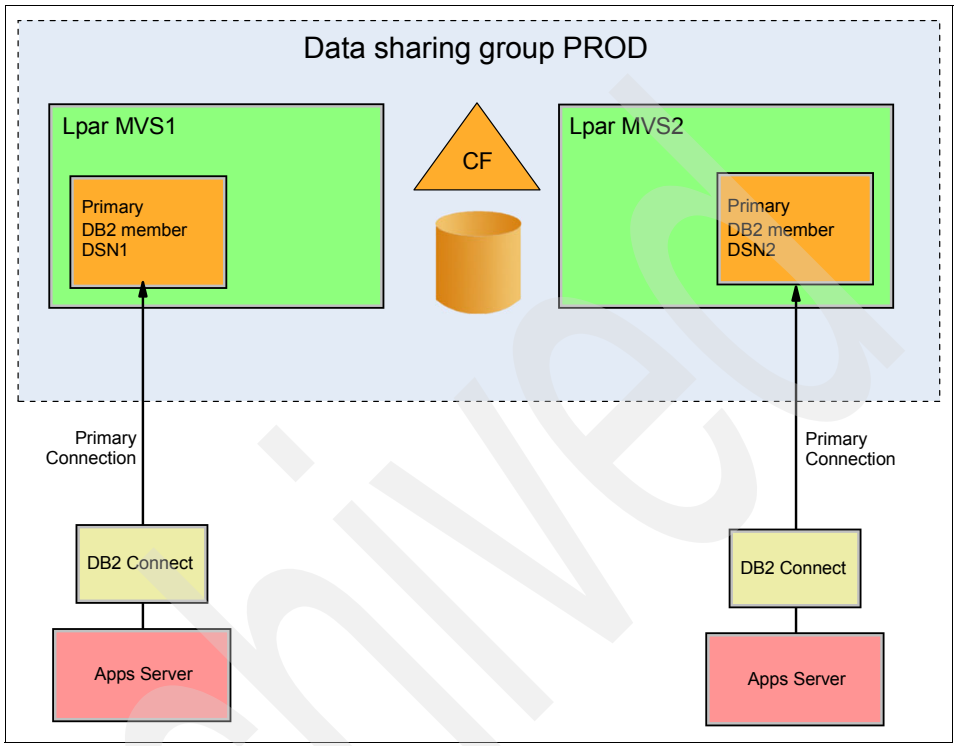



*Figure 1-4 Two active DB2 members without passive standby members*

This option is considered most often when one DB2 member running in one CEC or LPAR cannot handle the PeopleSoft database workload from a CPU capacity standpoint or the PeopleSoft database workload would consume the entire 2 GB of virtual memory in the DBM1 address space. The 2 GB virtual storage limit is true for DB2 releases V7 and earlier, although DB2 V7 enables you to implement dataspaces to take advantage of 64-bit real storage on zSeries machines.

When thinking about configuring DB2 to support more workload, this is most often the first thought that comes to mind. In this configuration, PeopleSoft sysplex failover scenario is set up so that application servers will move (connect) from the failing active DB2 member to the other active DB2 member. If the respective machines supporting these DB2 members are sized just to fit the normal workload, then one should expect degraded performance when all of the PeopleSoft database workload is concentrated on the surviving DB2 member. This degraded performance could come about due to lack of CPU capacity or lack of memory.

With recent announcements of the new zSeries z990 hardware and the latest version, V8.1, of DB2 software, both the CPU and virtual storage constraints are being lifted. With 64-bit addressing support, it might be possible to revert to the previous option and still support increased PeopleSoft workload.

In general, we do not recommend that this option be used. We recommend option 2 instead. If you decide to use this option anyway, then we remind you to give careful consideration to sizing the hardware properly and configuring Workload Manager (WLM). If you require the same level of performance, no matter what state the system is in, then each system should have enough CPU and memory capacity reserved to handle the maximum additional workload on each system. This standby capacity can be very costly if using the resources just for one PeopleSoft system. Fortunately, one of the great strengths of z/OS on zSeries is the capability to support multiple workloads simultaneously. This is where WLM is important. WLM enables you to assign importance to each workload. So in the event of a failover of workload to one surviving DB2 member, WLM can be configured to ensure that the PeopleSoft workload receives priority over the other workload, even if it is non-production PeopleSoft workload.

A real-life customer example of Option 1 is described in Appendix A, “PeopleSoft data sharing configuration (option 1): customer example” on page 111.

### *Two active DB2 members, each with a passive standby member in opposite LPAR - option 2*

Data sharing option 2 is a realistic implementation for a large PeopleSoft installation to allow database availability. With option 2, standby data sharing members are provided for each of two primary members, as shown in Figure 1-5 on page 10.

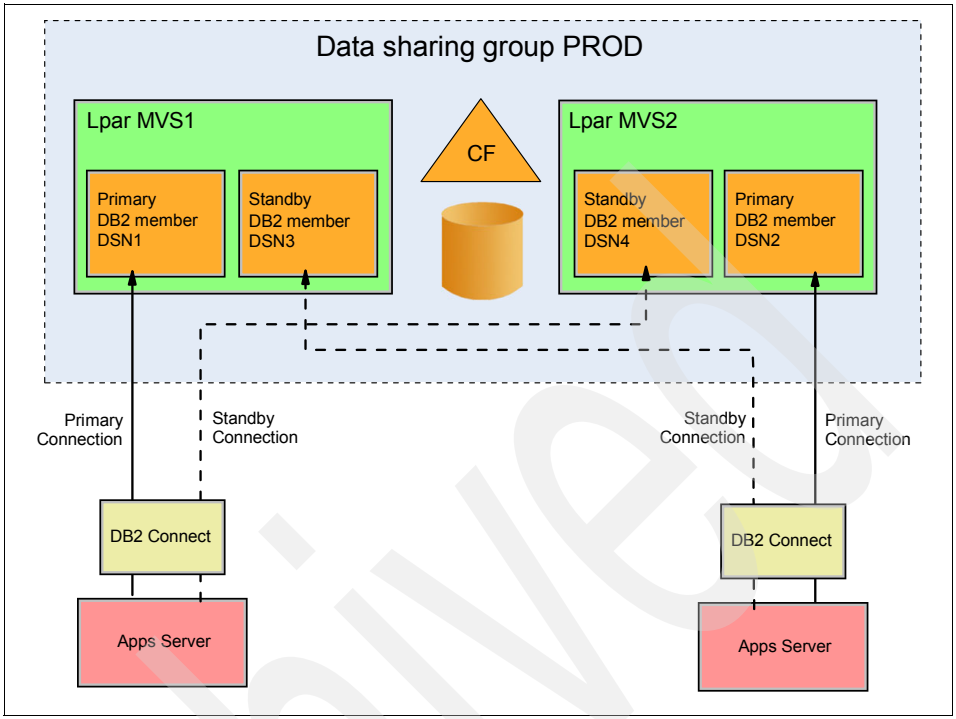



*Figure 1-5 Two active DB2 members, each with a passive standby member in opposite LPAR - Option 2*

Option 2 is really just a variation of option 0. In both options you have an active and standby DB2 member, but option 2 just has more pairs of active and passive members. This option is recommended (or even required) to support any PeopleSoft requirement that exceeds the capacity of a single machine.

Another use of this option would be to isolate PeopleSoft business components from each other. This is the logical extension of having separate application servers to run specific PeopleSoft modules.

As in the other options, it is possible that the MVS1 and MVS2 LPARs are performing other work, as long as the load is light enough to enable adequate performance if activation of one of the standby DB2 members occurs.

### *Two active DB2 members, each with a passive standby member in independent LPAR - option 3*

Option 3 represents the option 2 solution carried to the next level. Not only are there standby data sharing members, there are also standby LPARs in which the DB2 members reside (Figure 1-6 on page 11).

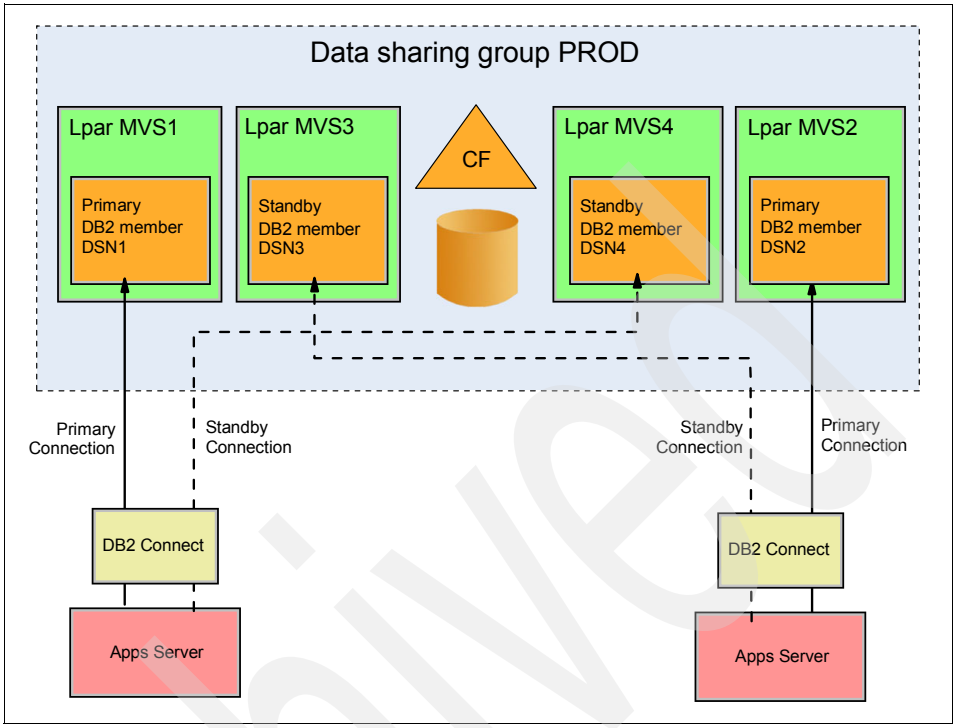



*Figure 1-6 Two active DB2 members, each with a passive standby member in independent LPAR - Option 3*

At first glance it may appear that additional LPARs for the standby function is a heavy price to pay for availability, but most installations that incorporate option 3 assign the standby function to one of the LPARs in their test-to-production landscape. That is, the normal function of MVS3 or MVS4 could be as a test or quality assurance machine, functions that can be interrupted if an availability situation arises.

However, the use of machines in the test-to-production landscape is not without risk; they could be using a level of z/OS or DB2 for z/OS that has not yet been put into production. Failover scenarios could then encounter situations with this new code that have software failures, operational differences, or performance impacts that are unexpected.

## 1.3.2 How many data sharing groups?

A typical PeopleSoft landscape consists of a development system, a quality control system, a stress test system, and a production system. Optionally, one

might decide to have a training system, a technical sandbox, or a production support system.

It is common these days for businesses to roll out the other PeopleSoft technology components such as PeopleSoft HR, Financials, and so on to support the next generation of PeopleSoft solutions. So it is quite common to have separate PeopleSoft landscapes for each PeopleSoft component.

Whatever PeopleSoft components or solutions you implement, the total number of PeopleSoft systems to build and maintain can add up quickly. It seems that every group involved in implementing a PeopleSoft system or solution wants their very own system to work with. Of all of those systems, how many should be configured for high availability and performance using data sharing?

As you are reading this book, you have already decided that your PeopleSoft production system should be configured to be highly available and/or scalable. Therefore, the production system must be configured at the very least to run in DB2 data sharing mode.

What about the non-production PeopleSoft systems? The answer is not so easy. It depends on your service level agreement (SLAs). Some SLAs require that even the development system be highly available. It is very costly to have developers who cannot work because the system is unavailable. Whatever your SLA, we recommend that you configure at a minimum one other PeopleSoft system for DB2 data sharing in your promote-to-production landscape. This system is where you would test your code or configuration changes to verify that there are no problems related to running them in your data sharing setup. Your production system is not the place to learn that your changes do not work with DB2 data sharing.

Which non-production system should be configured for data sharing? It depends on how soon or late you want to test your changes with data sharing. Applications will run just fine with data sharing when doing single-user or component-level tests. The story might be quite different for stress tests. As the number of users running against different systems increase, you might have a bigger potential for resource contention, so we recommend that your other data sharing system is either your quality control system or your stress test system if you have one.

We recommend further that you consider having at least one additional data sharing system in each of your PeopleSoft landscapes where your business needs require that you have high availability for your production system. While each PeopleSoft component shares common technology, there does exist non-common functionality. A more important thing to keep in mind is specific landscape configuration work. So it is recommended to have one additional data sharing system per PeopleSoft Landscape.

So far we have concentrated on figuring the number of data sharing systems to make sure that the application changes and PeopleSoft system changes do not cause any problems with data sharing. What about the infrastructure changes like coupling facility changes? What system should the infrastructure group use to test their changes? The infrastructure group should consider building a Parallel Sysplex with data sharing system that is independent of the production and non-production PeopleSoft systems. It is sufficient to have one Parallel Sysplex for the infrastructure group. There is no need nor benefit to have one technical sandbox system per PeopleSoft landscape. This approach, while consistent, would be cost-prohibitive.

### 1.3.3 How many data sharing members?

When you have decided that you need DB2 data sharing, the next question is how many data sharing members are required for each highly available system. The answer to this question depends on the data sharing option you are implementing. The option you choose depends on the sizing estimate for your proposed production system or systems.

For an option 0 production system, you need only two data sharing members per data sharing group. The primary data sharing member does all of the work, and the secondary data sharing member is a standby only. We call this passive data sharing. This option is valid as long as your workload does not exceed the PeopleSoft rating of the zSeries box or production LPAR. One fact about this configuration that is difficult to live with is that you have a second zSeries box or LPAR of at least equal CPU capacity, memory capacity, and I/O bandwidth sitting there using electricity, but not doing any active work. So that leads us to option 1.

For an option 1 production system you have active workload distributed between two or more members of a data sharing group. Assume that you are configuring a two-way data sharing system: If one system fails or becomes unreachable, the workload will be moved to the surviving data sharing member. If you want the system to perform in failover mode with same level of throughput as in non-failover mode, then you must ensure that there will be sufficient extra CPU capacity, memory capacity, and I/O bandwidth to handle the failover work in one zSeries box or LPAR. Basically, you must double the capacity of each zSeries box. A zSeries box rarely fails, so it may not be so important to have all of that extra capacity ready for a failover situation that will rarely happen. In the DB2 V7.1 and earlier releases, keep in mind that there is a 2 GB limit on the addressable memory in the DBM1 address space. Every PeopleSoft work process that connects to the surviving data sharing member will consume from 1.3 to 2.5 MB (or even higher). So you must plan for the maximum number of RRSAF connections per data sharing member.

Another possible option 1 production system could have three data sharing members in a group. If one data sharing system fails with this version of option 1, you have the option to redistribute the workload to one of the surviving members or to redistribute the workload evenly to the surviving members. When making this decision, the main choices are to minimize DB2 inter-systems interest or not overallocate DB2 DBM1 virtual storage. To minimize DB2 inter-systems interest, ideally you would move all of the workload from the failing data sharing member to one of the surviving data sharing members. This might lead to overallocation on DBM1 virtual storage. On the other hand, to prevent possible overallocation of memory, you could evenly distribute the workload among the surviving members. However, this might increase DB2 inter-systems interest between the two surviving members. Which is the best choice to make? It is better to have the system available providing reduced throughput than to overallocate DBM1 virtual storage and risk another abend that would reduce throughput even more. Therefore, we recommend for option 1 systems redistribution of the workload evenly among the surviving data sharing members.

In order to minimize DB2 inter-systems interest and prevent the overallocation of memory in a failover situation, we recommend that you implement option 2 instead of option 1. Option 2 eliminates DB2 inter-systems interest, because all workload from the failing DB2 data sharing member would move to a standby member. Also, no part of the surviving primary data sharing members would be affected. This standby member could be started, ready, and waiting in the same LPAR as the primary or in another LPAR. Option 2 eliminates the possibility of overallocation of DBM1 virtual storage, because the failover happens to an empty data sharing member that is configured exactly the same as the primary. In the event that only some of the PeopleSoft work processes fail over to the standby data sharing member, a corresponding number of PeopleSoft work processes would be eliminated from the primary data sharing member. There would be no overallocation of virtual storage, but there could be inter-systems interest, but only for that portion of workload running on those data sharing members.

We recommend that you configure your non-production data sharing system or systems exactly the same way as your production system.

The notion of standby DB2 members was introduced for several reasons: less disruptive to surviving DB2 members (no washing of the buffer pools or dynamic statement caches) and it reduces the need to ensure that there is sufficient DBM1 virtual storage to absorb the movement of a large number of PeopleSoft work processes (and hence DB2 threads). There are two types of companies doing PeopleSoft-DB2 data sharing: those who are motivated primarily by near-continuous availability (secondarily the ability to level workload across the multiple CECs required for no single point of failure) and those who have both a scalability and availability objective.

The first type typically implements option 0 and frequently is characterized by having many PeopleSoft production systems.

The second type generally implements option 2. Many of the companies pursuing both scalability and high availability have three or more CECs in their sysplex. A typical configuration in a three-CEC sysplex would have a primary DB2 on each CEC with a standby DB2 member residing in each production LPAR that contains a primary DB2. In the event of planned or unplanned loss of one of the production environments (such as CEC1), half of the application servers will reconnect to the standby DB2 member on CEC2, and half will connect to the standby member on CEC3. Although this is more complex than simply moving all of the application servers to one standby DB2, it does offer workload management benefits. Assuming that each of the three primary DB2s were servicing one-third of the total workload, half of this (one-sixth of the total workload) will be moved to each standby member in the surviving CECs. When coupled with the Workload Manager's ability to differentiate priorities (goals, importance, velocity) based on PeopleSoft work process type, very high interactive service levels can be maintained with minimized disruption to the surviving CECs. This enables us to minimize the purchase of extra capacity to support failover.

2

# Performance considerations: implementing PeopleSoft with Parallel Sysplex data sharing

Our rationale for approaching this subject is to build a bottom-up case for good PeopleSoft performance. A well-performing sysplex is the base for a well-performing DB2 data sharing system, and a well-performing DB2 data sharing system is the base for a well-performing PeopleSoft system.

In this chapter we discuss performance issues in the order you would approach them at planning and implementation time:

- Sysplex performance
- Data sharing performance

We primarily focus on methodology and refer you, when necessary, to publications where you can find more detailed discussion on the subjects covered.

## 2.1 Sysplex performance

This section discusses sysplex performance and monitoring issues relevant to PeopleSoft data sharing environments. A performing sysplex that is well set up and healthy is the basis for good performance in your DB2 data sharing system.

To ensure that your tuning efforts are producing the expected results, you should keep a baseline for a before-and-after comparison. This baseline may consist of:

- Coupling Facility processor utilization
- Average synchronous and asynchronous service time
- Percentage of changed requests

We describe the following:

- CF performance monitoring and tuning
- Sysplex workload monitoring
- WLM policies
- RMF™ Monitor III
- RMF Post Processor reports
- RMF spreadsheet

### 2.1.1 CF performance monitoring and tuning

It is important to have a basic understanding of CF performance tuning concepts in order to have a properly tuned DB2 data sharing group. Three items should be monitored:

- CF processor monitoring
- CF storage utilization
- CF link performance

Monitoring CF utilization and performance can be achieved by using RMF Spreadsheet Reporter, which is described in 2.1.5, “RMF Spreadsheet Reporter” on page 28.

#### CF processor monitoring

CF processor utilization must be kept below 50% because synchronous service time starts to degrade with higher utilization. High sync service time may have a performance impact because the requestor’s processor has to wait for the request to complete. Besides reducing CF efficiency, it also increases CPU utilization in the z/OS side.

Our recommendation is to keep CF utilization under 30%, if you are in a single CP Coupling Facility, in case a failover happens and one CF has to absorb the workload from a failing CF.

To obtain information about CF utilization, run an RMF Post Processor report with option SYSRPTS(CF).

### CF storage utilization

CF level 12 introduces 64-bit mode support and removes the 2 GB limitation that existed for some DB2 structures. CF level 12 provides the following advantages:

1. 64-bit CFCP with CFLEVEL12 and OS/390 R10 to overcome 2 GB control storage limit and provide for very large LOCK1 structures (because of modified record list requirement)
2. 64-bit real with OS/390 R10 and any z/OS operating system to support dataspace buffer pools and provide VSCR above line in DBM1
3. 64-bit virtual support with z/OS R3 and DB2 for z/OS V8 to provide even more VSCR
4. Reduced data sharing overhead using WARM/RFCOM with z/OS R4, CFLEVEL12, and DB2 V8
5. SLPIT Recovery (BACKUP SYSTEM utility and RESTORE SYSTEM utility without LOGONLY option) with z/OS R5 and DB2 V8

Special care also must be taken in anticipation of failover scenarios. The surviving CF must have enough capacity to absorb the structures coming from a failing CF.

### CF link performance

Service time is monitored from the moment an exploiter issues a CF request to the moment z/OS receives the command return. Service time is recorded in microseconds for each structure used by each system. As CF links can be shared between two or more LPARs in the same CPC, it is possible that some contention exists.

When such a contention occurs, one request is processed and the other is rejected. This situation gets registered as `Path busy`. It is important to check the total percent of delayed requests in the Subchannel Activity section from an RMF CF report.

If the total path busy is greater than 10% of all requests, this indicates link contention, consider dedicating CF links to the CF from each partition or adding more shared links to the CF.

Each CF link is capable of supporting a certain number of concurrent CF operations. The maximum number of concurrent CF operations is the total number of subchannels (buffers) defined to a CF.

When all subchannels are busy performing operations to the CF, then newly incoming requests are delayed. This is known as a *subchannel busy* condition. For SCA and Group buffer pool requests, z/OS will queue these delayed requests and process them in first-in-first-out order as soon as one of the active operations ends. For a lock structure, z/OS will generally spin and wait for one of the active requests to finish and then re-issue the request.

In summary, for SCA and Group buffer pools, increased subchannel busy may lead to request queuing, thus affecting performance. For the Lock structure, high subchannel busy leads to higher CPU utilization and diminished CF efficiency.

If the total subchannel busy is greater than 10% of all requests, you may have to add another CF link between the CF and the z/OS partition.

We recommend monitoring service times. The zSeries family of processors introduced three additional links: ISC-3, ICB-3, and IC-3. These links must be configured in Peer Mode. Peer Mode supports CF between z900 servers and provides both sender and receiver capability on the same link. Peer links provide up to seven expanded buffer sets (subchannels), which in reality become up to 14 buffer sets as both receiver and sender traffic travel in the same link.

Each link type has its own characteristics. They facilitate improved channel efficiency and service times in the processing of CF requests. But it is beyond the scope of this book to provide more in-depth details on CF links.

Service time varies according to the hardware involved. Figure 2-1 shows the guidelines for service times for the z/900 series:

```
Synchronous service time:
        GBPs            LOCK1, SCA
ICP     30 mics         20 mics
CFP     65 mics         40 mics

Asynchonous service time:
        GBPs            LOCK1, SCA
ICP     120 mics        100 mics
CFP     150 mics        120 mics
```

*Figure 2-1 Service times for z/900 series*

Starting with z/OS 1.2, IBM has introduced a new algorithm to decide whether to convert synchronous requests to asynchronous requests. This new function monitors CF service times for all (CACHE, LIST, and LOCK) structures and, based on new internal thresholds, determines which requests would be more efficiently executed asynchronously. Thresholds are different for simplex and duplex requests and for Lock and non-Lock requests.

For more insight, review Washington System Center Flash # 10159. You can download it from:

http://www-1.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130103.pdf

## CF signalling path length monitoring

z/OS systems within a sysplex must constantly pass messages to each other. These signalling paths are heavily used when IRLMs have to negotiate locks in a data sharing environment.

When Global Lock contention is detected, Global Lock Management via XES/IRLM uses message passing and is heavily dependant on fast XCF communications. Global Lock Management is synchronous with respect to the application. The lack of enough XCF signaling resources to accommodate Global Lock contention management traffic can be detrimental to DB2 performance. It is best to overconfigure XCF signaling resources.

We have observed that low-weight LPAR partitions can get worse performance for anything dependent on asynchronous operations and XCF messaging (such as Global Lock contention). There are several spots they can slow down:

- The z/OS dispatcher must run to see an asynchronous completion. The LPAR may not have any logical CPs for MVS1D dispatched when the asynchronous operation completes, so MVS1D is very late in noticing.
- Several SRBs must run to complete the asynchronous operation, post XCF, and then wake up IRLM. LPAR can take away the logical processor running any of this work any time. This could also slow down lock contention on other partitions, as they sometimes have to wait for MVS1D to respond to them

Transfer time between LPARs can only be seen by issuing the commands D XCF,PI,DEV=ALL,STATUS=WORKING and D XCF,PI,STRNM=ALL. The recommended transfer time is less than 2000 microseconds (or 2 milliseconds). These IRLM negotiations are synchronous, so keeping them under 2000 microseconds is very important.

It is recommended that these two commands be issued automatically every hour.

To monitor signalling paths, you must run an RMF postprocessor with REPORTS(XCF) control card. For XCF tuning guidelines, refer to IBM Washington Center Flash # 10011 at:

http://www-1.ibm.com/servers/eserver/zseries/library/whitepapers/pdf/availchk_parsys.pdf

This flash provides a step-by-step road map to ensure that transfer time between LPARs is less than 2000 microseconds.

Figure 2-2 shows an example of an RMF postprocessor output from option REPORTS(XCF).

```
 ----- BUFFER -----      ALL
TO        TRANSPORT  BUFFER        REQ   %    %    %    %     PATHS     REQ
SYSTEM    CLASS      LENGTH        OUT  SML  FIT  BIG  OVR  UNAVAIL  REJECT
3W02      DEFAULT    16,316    119,266  100    0   <1  100        0       0
          DEFSMALL      956    129,157    0  100    0    0        0       0
3W03      DEFAULT    16,316    197,102  100    0   <1  100        0       0
          DEFSMALL      956    242,876    0  100    0    0        0       0
3W05      DEFAULT    16,316    183,354  100    0   <1  100        0       0
          DEFSMALL      956    206,937    0  100    0    0        0       0
                                ----------
TOTAL                            1,078,692
```

*Figure 2-2 RMF postprocessor output from REPORTS(XCF)*

## 2.1.2 WLM policies

The purpose of Workload Manager (WLM) is to balance the available system resources to meet the demands of z/OS subsystems and applications based on user-defined performance objectives. WLM provides a mechanism for managing workload distribution, workload balancing, and distribution resources to competing workloads. It manages the workload of the entire z/OS system. This section focuses on some of the considerations for the PeopleSoft application.

WLM is a critical success factor. We strongly recommend that you configure a WLM policy specific to PeopleSoft to help you manage the performance of your PeopleSoft system and achieve good overall system performance for your sysplex.

WLM service definition contains all of the information about the installations needed for workload management processing. There is one service definition for the entire sysplex. The service level administrator specifies the service level definitions for the entire sysplex. The service level administrator sets up *policies* within the service definition to specify the goals for work. A service level administrator must understand how to organize work and be able to assign it performance objectives.

Is it important that the PeopleSoft applications be included in the WLM service definition. The PeopleSoft applications should be viewed as a multi-application subsystem within the z/OS subsystems and applications. Because of the variety of workloads that can be run within PeopleSoft, proper performances should be established within the PeopleSoft suite. In general, you should consider giving

the online PeopleSoft applications higher performance goals than the PeopleSoft batch workloads.

When the PeopleSoft application suite performance goals are established, the PeopleSoft workload definition must be blended with the total z/OS workload.

Another consideration for the WLM service definition is to define reporting classes for each PeopleSoft process type. This provides the greatest granularity for monitoring purposes. The Resources class definition included in Example 2-1 shows how this can be accomplished.

*Example 2-1 Sample setup for WLM and PeopleSoft V8*

```
1 SI         PRODDB2                                DDFMED   PRODDB2
           2 UI       PRHRID                                DDFMED   PSHRDFLT
             3 . CI       . cr*                             PSOFTSQP PRHRCR
             3 . CI       . CR*                             PSOFTSQP PRHRCR
             3 . CI       . db2b*                           PSOFTSQP PRHRDB2B
             3 . CI       . javaw*                          PSOFTSQP PRHRJAVW
             3 . CI       . PSAE*                           PSOFTSQP PRHRPSAE
             3 . CI       . psae*                           PSOFTSQP PRHRPSAE
             3 . CI       . PSCR*                           PSOFTSQP PRHRPSAE
             3 . CI       . pscr*                           PSOFTSQP PRHRPSAE
             3 . CI       . PSDAEMON                        PSOFTSQP PRHRDAEM
             3 . CI       . psdaemon                        PSOFTSQP PRHRDAEM
             3 . CI       . PSAPPSRV                        PSAPPSRP PRHRAPPS
             3 . CI       . PSQCKSRV                        PSAPPSRP PRHRQCKS
             3 . CI       . PSSAMSRV                        PSAPPSRP PRHRSMAS
             3 . CI       . PSQRYSRV                        PSAPPSRP PRHRQRYS
             3 . CI       . PSBRK*                          PSAGENTP PRHRBRK
             3 . CI       . PSMSG*                          PSAGENTP PRHRMSG
             3 . CI       . PSPUB*                          PSAGENTP PRHRPUB
             3 . CI       . PSSUB*                          PSAGENTP PRHRSUB
             3 . CI       . PSPRCSRV                        STCMD    PRHRSCHD
             3 . CI       . PSDSTSRV                        PSAGENTP PRHRSCHD
             3 . CI       . PSQE*                           PSOFTSQP PRHRQRY
             3 . CI       . psqe*                           PSOFTSQP PRHRQRY
             3 . CI       . PST*                            PSAPPSRP PRHRPST
             3 . CI       . pst*                            PSAPPSRP PRHRPST
             3 . CI       . PSS*                            PSAPPSRP PRHRTR2
             3 . CI       . pss*                            PSAPPSRP PRHRTR2
             3 . CI       . PSDM*                           PSOFTSQP PRHRPSDM
             3 . CI       . dsdm*                           PSOFTSQP PRHRPSDM
             3 . CI       . PSD*                            PSAPPSRP PRHRTRD
             3 . CI       . SQR*                            PSOFTSQP PRHRSQR
             3 . CI       . sqr*                            PSOFTSQP PRHRSQR
             3 . CI       . PSIDE*                          PSAPPSRP PRHRIDE
             3 . CI       . pside*                          PSAPPSRP PRHRIDE
             3 . CI       . PSNV*                           PSOFTSQP PRHRNVS
```

```
        3 . CI       . psnv*                        PSOFTSQP PRHRNVS
        3 . CI       . PS*                          PSOFTSQP PRHRDDF
2 UI       PRFIID                          DDFMED   PSIFDFLT
        3 . CI       . cr*                          PSOFTSQP PRFICR
        3 . CI       . CR*                          PSOFTSQP PRFICR
        3 . CI       . db2b*                        PSOFTSQP PRFIDB2B
        3 . CI       . javaw*                       PSOFTSQP PRFIJAVW
        3 . CI       . PSAE*                        PSOFTSQP PRFIPSAE
        3 . CI       . psae*                        PSOFTSQP PRFIPSAE
        3 . CI       . PSCR*                        PSOFTSQP PRFIPSAE
        3 . CI       . pscr*                        PSOFTSQP PRFIPSAE
        3 . CI       . PSDAEMON                     PSOFTSQP PRFIDAEM
        3 . CI       . psdaemon                     PSOFTSQP PRFIDAEM
        3 . CI       . PSAPPSRV                     PSAPPSRP PRFIAPPS
        3 . CI       . PSQCKSRV                     PSAPPSRP PRFIQCKS
        3 . CI       . PSSAMSRV                     PSAPPSRP PRFISMAS
        3 . CI       . PSQRYSRV                     PSAPPSRP PRFIQRYS
        3 . CI       . PSBRK*                       PSAGENTP PRFIBRK
        3 . CI       . PSMSG*                       PSAGENTP PRFIMSG
        3 . CI       . PSPUB*                       PSAGENTP PRFIPUB
        3 . CI       . PSSUB*                       PSAGENTP PRFISUB
        3 . CI       . PSPRCSRV                     STCMD    PRFISCHD
        3 . CI       . PSDSTSRV                     PSAGENTP PRFISCHD
        3 . CI       . PSQE*                        PSOFTSQP PRFIQRY
        3 . CI       . psqe*                        PSOFTSQP PRFIQRY
        3 . CI       . PST*                         PSAPPSRP PRFIPST
        3 . CI       . pst*                         PSAPPSRP PRFIPST
        3 . CI       . PSS*                         PSAPPSRP PRFITR2
        3 . CI       . pss*                         PSAPPSRP PRFITR2
        3 . CI       . PSDM*                        PSOFTSQP PRFIPSDM
        3 . CI       . dsdm*                        PSOFTSQP PRFIPSDM
        3 . CI       . PSD*                         PSAPPSRP PRFITRD
        3 . CI       . SQR*                         PSOFTSQP PRFISQR
        3 . CI       . sqr*                         PSOFTSQP PRFISQR
        3 . CI       . PSIDE*                       PSAPPSRP PRFIIDE
        3 . CI       . pside*                       PSAPPSRP PRFIIDE
        3 . CI       . PSNV*                        PSOFTSQP PRFINVS
        3 . CI       . psnv*                        PSOFTSQP PRFINVS
        3 . CI       . PS*                          PSOFTSQP PRFIDDF
```

The example shows that HR and FI are separated by a UI classification rule. This type of rule can also be used to separate production HR from test, dev, and so forth.

Also notice that service classes are shared by many CIs but report classes are mostly unique.

Finally, there are cases where CIs come through in lower case; therefore, both cases have to be specified.

Here are some samples for the service class definitions, assuming that the service coefficients are set to a multiplier of 1:

```
PSAGENT - single service class, importance 3 or 4, velocity of 30
PSAPPSRP - two period service class:
   period 1 - importance 1, duration 500, velocity 50
   period 2 - importance 3, velocity 30
PSOFTSQP - two period service class:
   period 1 - importance 3, duration 10000, velocity 40
   period 2 - importance 4, velocity 30
```

The key to establishing the proper WLM service definition is to understand the entire workload on the system, then to monitor the system through detailed reporting class definitions and make the necessary adjustments as the workload demands change.

### 2.1.3 RMF Monitor III - Work Delay Monitor

The RMF Work Delay Monitor, otherwise known as RMF Monitor III, provides very useful reports for identifying delays in the system components that have an impact on your PeopleSoft workload performance:

- CF activity overview

  From your ISPF RMF Monitor III menu, select **S SYSPLEX** → **5**.

  This displays an overview of your coupling facility activity. Check that the CPU activity is no higher then 50% and that there is free storage.

- CF paths

  From your ISPF RMF Monitor III menu, select **S SYSPLEX** → **6**.

  This displays availability and performance of the paths to the CF. Check that you have all defined paths available and that the delay % is not too high.

- Delay Report

  From your ISPF RMF Monitor III menu, select **1** → **4**.

  This takes you to the Delay Report. A shortcut to this report is to enter `DLY` from the command line on any RMF Monitor III screen.

  Delay Report is a very useful report for identifying delays to your PeopleSoft system workload components.

The two main types of delay you are likely to see are PRC and DEV:

– PRC

  PRC indicates a delay for processor resources. If you have not set up your WLM service classes correctly, then you may have less important work getting preference for CPU resources over and above your DB2 or PeopleSoft workloads.

  One cause of PRC delays might be that the LPAR is reaching its allowable CPU limit. You should check the PR/SM™ weightings and the CPU activity for the machine. The CPU activity can be checked using the RMF post processor reports. You can also see the machine CPU activity in real time from the RMF Monitor III. To see the machine CPU activity from Monitor III do the following:

  From your ISPF RMF Monitor III menu, select **1** → **3**.

  This option shows the CPU activity for the whole machine and for the LPARs defined on the machine.

– DEV

  This indicates a delay for a DASD or tape. For your DB2 address spaces, you may see DASD device delays. This can happen for several reasons, the most common being that there are too many active tablespace datasets on a single volume. This can be related directly to your PeopleSoft database response times. If you have high access times for a table, as identified in your PeopleSoft single record statistic analysis or your PeopleSoft SQL trace analysis, then this may be caused by device delays in DB2 for the volume.

  In the DB2 system overview, you might also see high I/O suspend times.

  You can investigate further by selecting the resource device delay report as follows:

  - From your ISPF RMF Monitor III menu, select **3** → **2**.

    This shows disk volumes that are causing delays to the current workload. You can get further detail on the datasets that are being accessed on the volume.

  - From your ISPF RMF Monitor III menu, select **O** → **4** → **Report=DSNV**.

    Now enter the volser of the volume you want more details about.

    Return to the ISPF RMF Monitor III menu.

- From your ISPF RMF Monitor III menu, select **3** → **3B**.

  This produces the detail dataset delay report for that volume. You will see which datasets and therefore which tablespaces are on the volume that are being accessed.

Device delays can be reduced with the Parallel Access Volume (PAV) facility. PAV, in either dynamic or static form, can make a big difference in terms of I/O response times by significantly reducing IOSQ time while queuing on the UCB. PAV involves creating aliases for the UCB to reduce queuing and to push it down where possible to DASD controller. Use of PAV can significantly reduce (not eliminate) the need for careful dataset placement.

If you cannot use PAV, then you should consider reducing I/O contention by careful dataset placement. Even if you have implemented PAV you should still think about controlling dataset placement to achieve optimal performance and to use your DASD resources to their full potential. Dataset placement can be achieved by implementing SMS policies to ensure separation of data across volumes.

Another way to reduce I/O delays is to partition the tablespace to spread the I/O across volumes. Even if you have PAV you should still consider controlling the dataset placement for the partitions by using SMS policies or manual methods. You should keep in mind that OS/390 z/OS queues I/O activity at the UCB for the volume, despite that fact that you may be using new technology DASD hardware.

### 2.1.4 RMF post processor reports

RMF post processor reports provide historical performance data for your PeopleSoft data sharing environment. This section highlights some of the reports that you can use to analyze performance problems in your PeopleSoft data sharing environment. This is not an in-depth discussion of RMF reporting, but for PeopleSoft consultants who want to investigate performance in a data sharing environment, it is useful to be aware of the additional OS/390 and z/OS reporting facilities.

Further documentation on running these reports and reading the output can be found in the RMF Report Analysis manual for your operating system release.

- CPU Activity Report

  This report is useful for investigating CPU resource delays in your system. It can provide historical information about logical and physical processor utilization in your sysplex.

- Coupling Facility Activity Report

  This report provides useful information about each coupling facility attached to your sysplex. If your previous investigations lead you to believe that the performance problems are due to poor data sharing performance, then this report provides useful information about the activity in the coupling facility.

- XCF Activity Report

  This report provides information about the activity of your cross-system coupling facility. This is useful if you suspect that poor performance may be due to poor data sharing performance.

- Device Activity Report

  In this report one of the important items is to check whether PAVs are being distributed properly as needed. Either dynamic or static form can make a big difference in terms of I/O response times by significantly reducing IOSQ time on the UCB queuing.

  PAV involves creating aliases for the UCB to reduce queuing and to push it down where possible to the DASD controller. Use of PAV can significantly reduce (not eliminate) the need for careful dataset placement.

### 2.1.5 RMF Spreadsheet Reporter

RMF Spreadsheet Reporter is a set of applications comprised of Excel macros that can be downloaded to your workstation. RMF Spreadsheet Reporter extracts information from a server mainframe RMF Postprocessor report and presents this data graphically. This tool can be downloaded for free from:

http://www-1.ibm.com/servers/eserver/zseries/zos/rmf/rmfhtmls/rmftools.htm

RMF Spreadsheet Reporter consists of four applications:

- Collector
- Extractor
- Converter
- Several spreadsheets

After installing the product, you might need to customize the provided skeleton JCL in C:\RMFPP\PROGS\rmfpp.jcl. In some cases, you may want some customization that is not part of Collector.

RMF Spreadsheet Reporter enables you to submit an RMF Postprocessor job from your workstation into a server mainframe through TCP/IP.

### Collector

This module enables you to provide all information to build the job using the previously mentioned skeleton JCL. It enables you to choose the date and time, duration interval, and type of reports, and at job submission time provides two options for receiving the listings, Immediate and Deferred:

- Immediate

  Immediate starts your job and waits for its conclusion. If you are reading SMF buffer data from RMF, it is likely that your job will end in less than 10 minutes because it reads data that requires no sorting. Be careful because the default FTP value for JESPUTGETTO is 600 seconds. This parameter specifies how long an FTP-job-submitter can wait on the job until it is finished.

- Deferred

  Deferred is more appropriate because you need to read from SMF tape files and sort it. Your job is submitted and, when it ends, the output can be sent to your workstation via FTP.

### Extractor

When the result set from your batch job is sent to your workstation, you can run the Extractor. It creates a Report-Work-Set from the downloaded file.

### Converter

The Converter takes the output from the Extractor and converts the RMF reports to a spreadsheet format.

### Spreadsheets

The Excel macros are the core of RMF Spreadsheet Reporter. There are several type of macros. For our tuning purposes, we recommend:

- Summary report

  A graphically represented summary report for CPU utilization and DASD I/O rate.

- DASD activity report

  Graphically represented detailed I/O report.

- Workload Activity Trend in goal mode report

  Graphically represented detailed report on Service Class and so on.

- Coupling Facility trend report

  Detailed data for all CF structures, graphical reports on CF utilization, number of requests, service times for all LPARs, percentage of changed requests due to path busy or subchannel busy. Highly recommended.

RMF Spreadsheet Reporter is a great jump forward in assisting the whole mainframe server (such as z/OS and DB2) in the areas of creating baselines, tuning, and tracking performance.

## 2.2 Data sharing performance

This section discusses DB2 data sharing performance and monitoring issues that are relevant to PeopleSoft implementations in a sysplex environment. A well-configured data sharing environment is the basis for good performance in your PeopleSoft system. We describe the following:

- Local buffer pool performance
- Group buffer pool performance
- Dynamic SQL Cache
- Lock performance
- DB2PM statistics batch report
- Table space partitioning
- Managing DBM1 virtual storage
- Configuring DB2 sort pool and workfiles
- DB2 backup policies; avoiding disruptive backups

Having a baseline is also very important. This is the only way to ensure that improvements derived from DB2 system tuning can be quantified.

When we tune buffer pools our final objective is to reduce the number of pages read in or I/O traffic. I/O savings can be not only quantified but also compared to MIPS consumption through CPU time savings. Buffer pool hit ratios may be good performance indicators, but I/O traffic is a much better tracking metric.

We have used such a baseline for trend reporting as well. The suggested baseline can be a 24-hour period updated daily. At the beginning of the week, this baseline, which can be in a spreadsheet, can be published to all involved and management. It can be a powerful decision support tool.

The baseline can be comprised of:

- Total number of synchronous I/O reads for all local buffer pools
- Total number of pages read sequentially for all local buffer pools (pages read via seq. prefetch + pages read via list prefetch + pages read via dynamic prefetch)
- Global and Local Dynamic SQL Cache hit ratios
- Global locking contention rates: Real and False
- Average CPU consumption

The first three items come from a DB2PM Statistics report, and the rest can be obtained from an RMF Postprocessor CPU report.

In large environments, browsing different performance reports daily and acting in a pro-active manner can be time-consuming. But with such a spreadsheet, we usually have an idea of the overall health of our DB2 systems.

For instance, any 20% increase in I/O traffic (sync or sequential) for more than a day or two and even if the average DB response time is not being affected, warrants a more detailed investigation.

This baseline is also good to help you spot changes in the application behavior that you may not be aware of otherwise. For example, sometimes new SQL challenged code is promoted into production without the DB2 systems group being aware of it. You should be able to detect it, as I/O traffic is increased by this change.

### 2.2.1 Local buffer pool performance

One of the most important points when designing DB2 buffer pools for a PeopleSoft environment is to group similar objects according to their characteristics, such as:

- Large workset, highly updated, random access
- Small workset, highly updated, random access
- Large workset, little updated, random access
- Small workset, little updated, random access
- Highly updated, sequential access
- Little updated, sequential access

When tuning DB2 buffer pools the primary goal is to reduce or eliminate I/O by finding needed pages in the local buffers and to minimize synchronous single page I/O.

The recommendations for buffer pool tuning are geared toward local buffer pools with a mixed load of random and sequential pages.

You can find the lowest I/O rate for a buffer pool through an estimated page residency time.

If you were to plot buffer pool miss rates versus buffer pool size, the result would be a queuing (exponential) curve. The objective is to avoid getting below the

knee of the cursor, and the recommendation for VP+HP is to get you well beyond the knee and onto the flat part of the curve. The two general rules are:

1. Pages should be resident in the virtual pools for at least a minute.

   or

2. Pages should be resident in the combination of virtual and hiperpools for approximately 15 minutes.

Take several reasonably large, typical sample intervals of 15 minutes each and perform these two calculations:

1. Vpool residency = (# of vpool buffers / hiperpool writes per second)

   The result should be approximately 60 seconds. This formula can tell how long a page can be in a virtual pool before it is written to a hiperpool.

   If the average page residency is much greater than 60 seconds, you can cut back on vpool buffers and give more to hiperpool buffers (at least two hiperpool buffers for each vpool). To get the hiperpool writes, add up synchronous and asynchronous data mover hiperpool writes. This result tells you the number of pages moved out of the vpools by the least recently used algorithm.

2. Total (vpool + hpool) residency = (# of vpool + hpool buffers / pages read per second)

   This combined result should be approximately 15 minutes, assuming you have vpools and hiperpools.

   If you are not using hiperpools, just calculate residency by using the formula vpool residency = (# of vpool buffers / # of pages read).

   Pages read means the total number of pages read via sequential prefetch + number of pages read via list prefetch + number of pages read via dynamic prefetch.

### 2.2.2 Group buffer pool performance

In this section we discuss:

- Caching pages in the group buffer pool (GBP)
- Calculating GBP hit ratios
- Castout efficiency

#### Caching pages in the GBP

There are mainly two requirements for a page to be cached in the GBP:

- The pages must have been updated in a local buffer pool.

- There must be inter-DB2 interest in the page and at least one member has read/write interest.

When a page set or partition becomes GBP-dependent, all updated pages in the local buffer pool are moved to the corresponding GBPs. Updated pages in the local pool are not always written synchronously to the GBP. Prior to V8, all GBP page writes were CF synchronous operations. Pages were written synchronously with respect to the application using *force at commit* protocol at commit. Pages were written asynchronous to the application using CF synchronous operations due to VDWQT/DWQT thresholds being reached, or system checkpoint coming due, or as part of page P-lock negotiation. Updated pages were also written synchronously during index leaf page split. In V8, DB2 uses new WARM CF commands (asynchronous CF command, multiple pages). Index leaf page split will also exploit WARM, and the number of GBP writes will be reduced to two.

Changed pages can be written to the GBP before the updated transaction is committed when:

- One of the local buffer pool thresholds (DWQT or VDWQT) is reached.
- The local buffer pool is short of available pages.
- A system checkpoint is issued. Such an event can clean out a local buffer pool before the commit.
- The same page is required for update by another system. This page writing into the GBP is part of page P-lock negotiation.

When such pages are written to the GBP, all copies of those pages that are cached in other members' buffer pools are invalidated. This process is called cross-invalidation (XI). Next time the application references those pages, they must be loaded from the GBP (or DASD) in a synchronous request.

Another main point to consider is Directory Reclaims. Group buffer pools contain directory entries that are used to maintain information about all pages in all local buffer pools.

Thus, having sufficient directory entries in the GBP optimizes cache structure usage.

However, without sufficient directory entries in the GBP, a new reference to a page in a local buffer pool requires that the oldest, least recently used directory entry in the GBP be reclaimed, provided that the old entry is not associated with a dirty or changed page in the GBP.

## Calculating GBP hit ratios

With the above information in mind, the formula to calculate group buffer pool hit ratio could be:

```
(SYN.READS(XI)-DATA RETURNED / (SYN.READS(XI)-NO DATA RETURN +
SYN.READS(XI)-DATA RETURNED))
```

`SYN.READS(XI)-DATA RETURNED` means that a synchronous CF read was issued because a local virtual buffer pool or hiperpool had a page marked as invalid. The page existed in the group buffer pool and was returned.

`SYN.READS(XI)-NO DATA RETURN` means a synchronous CF read was issued because a local virtual buffer pool or hiperpool had a page marked as invalid and no page was found in the group buffer pool. The recommended hit ratio is >= 70%, provided that the GBP has no problems with directory entries being reclaimed due to cross-invalidations.

Another recommendation is to check periodically for cross-invalidations due to directory entry reclaims.

Automate the issuance of the command every four hours:
`DIS GBPOOL(*) GDETAIL(INTERVAL) TYPE(GCONN)`

Figure 15-4 shows an example of how the response to the DISPLAY GROUPBUFERPOOL command can be recorded in the MSTR job log.

```
DSNB788I  -D3WP   CROSS INVALIDATIONS
              DUE TO DIRECTORY RECLAIMS              = 0
```

*Figure 2-3 GROUPBUFFERPOOL command display*

## Castout efficiency

Castout is the process of flushing pages from the group buffer pool to DASD. Keep in mind that there is no physical connection between the CF where group buffer pools reside and DASD. Therefore, the castout process involves reading the page from the group buffer pool into DB2's private buffer (not part of the buffer pool storage) and flushing the page from the private buffer to DASD.

Castout occurs when:

- The number of updated pages for a castout class queue exceeds a class threshold (CLASST) value.
- The total number of updated pages for a group buffer pool exceeds a group buffer pool threshold (GBPOOLT) value.
- The group buffer pool checkpoint is initiated.

- There is no more inter-DB2 read/write interest in the page set.
- The group buffer pool is being rebuilt, but the alternate group buffer pool isn't large enough to contain the pages.

### *Group buffer pool class castout threshold (CLASST)*

Each group buffer pool contains a fixed number of castout class queues. DB2 internally maps updated pages that belong to the same page sets or partitions to the same castout class queues. Because the number of castout class queues is limited, it is possible that more than one page set or partition will be mapped into the same castout class queue. This internal mapping scheme is the same across all sharing subsystems.

When DB2 writes changed pages to the group buffer pool, it determines how many changed pages are on a particular class castout queue. If the number of changed pages on a specified castout class queue exceeds the threshold, DB2 casts out a number of pages from that queue.

The default group buffer pool class castout threshold is 10, which means that castout is initiated for a particular page set or partition when 10 percent of the group buffer pool contains changed pages for the class.

### *Group buffer pool castout threshold (GBPOOLT)*

This threshold determines the total number of changed pages that can exist in the group buffer pool before castout occurs. The default group buffer pool castout threshold is 50, so when the group buffer pool is 50% full of changed pages, castout is initiated.

You can enforced the trickle writing of data by lowering the CLASST and GBPOOLT thresholds to 1% and 20%, respectively.

When monitoring group buffer pools, make sure that CLASST is being hit much more often than GBPOOLT.

### *Unlock castout*

The UNLOCK CASTOUT counter should always be significantly less than the PAGES CASTOUT counter. If It is not, then the castout write I/O is not being done efficiently. Look at the fields PAGES CASTOUT and UNLOCK CASTOUT in a DB2PM Statistics report. DB2 usually includes more than one page in the lock request to write pages to DASD. Therefore, UNLOCK CASTOUT should be less than or equal to PAGES CASTOUT. It will be much less if multiple pages are written per I/O. This ratio is a good indicator of how well the Castout process is performing.

The thresholds for GBPCHKPT, GBPOOLT, and CLASST prior to DB2 V8 were too high. In V8 they are set at 4, 25, and 5 respectively.

We recommend aggressive monitoring of such messages as DSNB319A, 325A, 228I, and 327I (out of storage in GBP).

We also recommend use of XES Auto Alter, provided that DB2 APAR PQ68114 is applied as an effective co-req.

XES Auto Alter has two distinct and maybe competing algorithms (in this order):

1. Structure Full avoidance
2. (Entry/Directory) Reclaim avoidance

FULLTHRESHOLD considerations (assuming current structure size is less than SIZE) are:

- If both number of elements (DB2 pages) and entries (directory) exceed FULLTHRESHOLD, then the structure size can be increased. If either one or the other (but not both) is above threshold, structure size is *not* increased and "juggling" occurs between entries and elements.
- Recommend setting `SIZE=INITSIZE+30%`, `FULL THRESHOLD=90`, and `MINSIZE=INITSIZE`.

### 2.2.3 Dynamic statement cache

PeopleSoft uses the global dynamic statement cache, which contains the skeleton copy of prepared SQL statements and is held in DBM1 inside the EDM pool or in an EDM dataspace. It is implemented by CACHEDYN=YES.

We recommend moving the global dynamic cache to a dataspace. This saves considerable storage within DBM1. We also suggest reviewing APAR PQ73624 before moving the global dynamic cache to a dataspace.

If your monitor does not provide global dynamic cache hit ratios, use this formula to calculate the global DSC hit ratio:

```
(PREP_STMT_CACHE_INSERTS - PREP_STMT_CACHE_REQUESTS)/PREP_STMT_CACHE_REQUESTS
```

The global DSC hit ratio should be greater than 95%.

Information about Dynamic Cache hit ratios can be found in the DB2PM Statistics report (Figure 2-4 on page 37).

```
DYNAMIC SQL STMT             QUANTITY  /SECOND  /THREAD /COMMIT
---------------------------  --------  -------  -------  ------
PREPARE REQUESTS              7110.00    11.85    28.55    2.00
  FULL PREPARES               1766.00     2.94     7.09    0.50
  SHORT PREPARES              5344.00     8.91    21.46    1.50
GLOBAL CACHE HIT RATIO (%)      75.16      N/A      N/A      N/A

IMPLICIT PREPARES                0.00     0.00     0.00     0.00
PREPARES AVOIDED                 0.00     0.00     0.00     0.00
STMT INVALID (MAXKEEPD)          0.00     0.00     0.00     0.00
STMT INVALID (DDL)               0.00     0.00     0.00     0.00
LOCAL CACHE HIT RATIO (%)         N/C      N/A      N/A      N/A
```

*Figure 2-4 DB2PM Statistics report: DSC hit ratios*

## 2.2.4 Lock performance

Reducing the amount of locks taken by an application improves the lock performance. In a data sharing environment, locking tuning should be done both at local and global levels.

In general, locking problems can be prevented by coding frequent commits in the applications to release the locks and avoid contention. Problems usually arise in user-written programs that do not have an appropriate commit frequency.

A high number of updates and deletes with few commits can accumulate a high number of locks. Currently IRLM can manage up to 8,000,000 locks concurrently. When this threshold is reached, additional requests for locks terminate abnormally with SQL code -904, resource unavailable.

We recommend coding `PC=YES` in IRLM address space. This allows up to 8M locks. DB2 V8 no longer supports PC=NO. IRLM V2R2 will run in 64-bit mode and support more locks.

### DB2 locking mechanisms

DB2 provides mechanisms for limiting the number of locks that are concurrently held. This is achieved using the system parameters NUMLKUS and NUMLKTS, and the tablespace attribute LOCKMAX.

The system parameter NUMLKUS sets a limit on the number of locks that any individual DB2 thread can hold. When that limit is reached, the program that accumulated these locks terminates with sqlcode -904. The maximum value for NUMLKUS is 2,097,152 (make sure that DB2 APAR PQ52651 has been applied) and it is recommended as an initial, first-cut value. Setting a lower value for

NUMLKUS helps detect offending programs earlier and it is especially recommended for test systems. Nevertheless, in most production systems (except Retail component) a lower value for NUMLKUS is acceptable, but it should not be lower than 500,000.

The system parameter NUMLKTS sets the default for the tablespace attribute LOCKMAX.

If the number of locks on a particular tablespace exceeds the LOCKMAX value, these locks will be replaced by a single tablespace-scope lock (note that the LOCKMAX is enforced on a per-thread, per-tablespace basis). This process is called lock escalation, and it can eliminate the cases of exhausting the IRLM and CF lock structures.

On the other hand, lock escalations do not necessarily address all lock contention problems. They can lead to deadlocks but it is very likely that the deadlock victim (the process that needs to back out) is a process for which the backout is least expensive. This addresses the issue of long backouts caused by the IRLM or CF lock structure exhaustions.

Another challenging aspect of lock escalations is in coming up with an optimal value for LOCKMAX. This is very customer-specific and depends on the particular workload and available central storage resources. If set too high, LOCKMAX allows the transactions and reports to hold too many locks, which leads to the previously described problems. If set too low, it causes too-frequent lock escalations. For example, with LOCKMAX too low, an application process that acquired more than the LOCKMAX locks could trigger a lock escalation, which in turn could not be successfully executed due to other applications holding non-compatible locks. After the timeout period, the escalation-triggering application process would receive a -913 timeout and it would need to rollback. Had the LOCKMAX value been higher, the application might have completed without lock escalation.

With NUMLKUS set to 2,097,152 we recommend an initial value of 1,000,000 for LOCKMAX. This value can be adjusted by the customer, depending on the site-specific considerations and NUMLKUS setting described previously.

Apart from enabling lock escalation, you should ensure that the IRLM and CF are sized to maximize the number of locks that can be held concurrently.

For the IRLM, specify `PC=YES` in the IRLM start-up procedure, and set the IRLM region size to `0` (zero).

## Global locking

Conceptually, all locks taken in a data sharing group are global locks. However, not all locks must be propagated to the lock structure in the coupling facility.

The number of locks that must be propagated to the CF lock structure for a page set or partition is determined by the number of DB2 members that are interested in the page set and whether their interest is read or write. The CF lock structure consists of two parts:

- The Lock table, also known as hash table.This table is made up of many lock table entries, also called as hash classes. When a lock required by a member needs to be made known to other members, it is registered in the Lock table.
- IRLM relies on the System Lock Manager (SLM) component of z/OS (also called XES) Inter-system locking services. Various IRLM lock levels can ONLY map to one of two XES lock levels. IRLM IS and S locks map to XES -S lock. IRLM U, IX, SIX, and X locks map to XES-X lock.

  For example, with two members holding IX locks on same table space. Both IRLM IX locks map to XES-X locks—hence lock conflict. XES detects contention and global lock contention processing invoked by IRLM to finally determine that IX is really compatible with IX and to grant lock request. This can be a problem with RELEASE(COMMIT) as used by PeopleSoft. There are some changes in DB2 V8 that will help. First, IRLM now maps IX L-locks to XES S and will grant IX parent L-locks locally when only IS or IX L-locks held on the object.

  Parent L-locks are still sent to Lock Structure but never cause contention for common conditions. IX remains incompatible with S because S parent L-locks must now map to XES X. There will be the additional overhead of global contention processing to verify that a pageset S L-lock is compatible with another pageset S L-lock. But this is a rare case.

  The majority of cases are IS - IS, IS - IX, and IX - IX, and hence there will be overall performance benefits. Second, child L-lock propagation is no longer based on the parent L-lock. It is based on the cached (held) state of pageset P-lock. If pageset P-lock negotiated from X to SIX or IX, then child L-locks are propagated. This provides reduced volatility. If P-lock is not held at the time of child L-lock request, child lock will be propagated.

  Parent L-locks no longer need to be held in retained state after DB2 failure (for example, a pageset IX L-lock is no longer held as a retained X lock). This is an important availability benefit.

### *Locking components*

In a data sharing environment, lock information is held in three different components:

- IRLM

  IRLM contains the most detailed lock information.

- XES

  XES is an important locking component. All lock information passed to XES is stored in XES. XES recognizes only two types of locks: S & X.

- CF lock structure

  The CF lock table also recognizes only two types of locks: S & X

### *Lock contention types*

There are three types of lock contentions:

- False contention

  False contention takes place when the same lock table entry is used to represent more than one resource. This inconsistency happens when the lock table is not sized properly and the number of lock table entries is inadequate. The global lock manager XES comes into play by getting information from all other XESs in the group that have locks for the resource.

- XES contention

  If XES determines that false contention does not exist, it checks whether XES contention applies. XES only recognizes two types of locks modes: X and S. If an XES tries to register an X-mode lock and such a lock is already registered by another member, it has to check with IRLMs to find out exactly what type of lock IRLM requires, because, for example, IRLM might actually need an IX-mode. At this point XES passes control to the IRLM contention exit to resolve the conflict. The exit checks the real need of IRLM and if it is not a real contention, it is called XES contention.

- Global contention

  Global contention includes all types of contention.

When contention (XES, false, or global) occurs, one of the XESs is known as the global lock manager to resolve the contention.

Throughout false and XES contention, there is extensive XCF signaling traffic between IRLMs to resolve the conflicts. This underlines the need to have XCF message traffic well tuned up as mentioned in “CF signalling path length monitoring” on page 21.

### *Monitoring lock contention rates*

Global contention rate under 3% is great. If the global contention rate is constantly above 5%, an investigation is warranted.

False contention rate should not be more than 1.5% of all requests. The final goal is to minimize XCF signalling traffic to resolve lock conflicts.

We recommend that you use RMF, not DB2PM, to calculate global and false contention rates. The reason is that the new z/OS 1.2 algorithm that converts synchronous CF requests to asynchronous CF requests has also caused the DB2 Instrumentation Facility to inflate the false contention rate.

To calculate global and false contention rates, run RMF Postprocessor with option SYSRPTS(CF). In the COUPLING FACILITY STRUCTURE ACTIVITY section, look for the lock structure data shown in Figure 2-5.

```
REQ TOTAL     2933K (A)
-CONT          456K (B)
-FALSE CONT     88K (C)
```

*Figure 2-5 Coupling facility lock structure activity*

`REQ TOTAL` represents the number of sync and async request to this lock structure.

`CONT` represents the number of lock requests that resulted in contention.

`FALSE CONT` represents the number of requests that resulted in false contention.

To calculate global contention rate, the formula is B/A. In the example in Figure 2-5, it would be 15.5%.

To calculate the false contention rate, the formula is C/A. In the case above, that would be 19.3%.

A noteworthy comment is that this formula will assist you if you have a uniform workload going across all members of the group. However, if that is not the case, you might need to check global and false contention on a LPAR basis.

### *Monitoring lock structure storage*

To help monitor the efficient use of CF lock structure storage, you can use the -DISPLAY GROUP command as shown in Figure 2-6 on page 42.

```
SCA   STRUCTURE SIZE:  80128 KB, STATUS= AC, SCA IN USE: < 1 %
LOCK1 STRUCTURE SIZE: 120064 KB
NUMBER  LOCK ENTRIES:  33554432 (A)
NUMBER  LIST ENTRIES:    174921 (B),LIST ENTRIES IN USE:1887(C)
```

*Figure 2-6 DISPLAY GROUP command showing the CF lock structure*

Where (A) is the maximum number of lock entries possible for the lock table, (B) is the maximum number of modify lock list entries, and (C) shows how many of those list entries are in use.

The coupling facility lock structure size limits how many locks can be held concurrently. CFLEVEL 12 provides some relief because it enables the lock structure to reside beyond the old 2 GB storage limitation.

Storage allocated by the lock structure is comprised of two parts:

- Lock hash table

  This is also called the lock table or the hash table. The hash table is where DB2 members register interest in a resource. The hash table is comprised of many table entries or hash classes. A lock needed by a DB2 member is registered in the hash table when that member needs to make it known to other members in the group. If this hash table is too small, locks might be represented by a single value. This introduces false contention, as two locks for two different resources might be represented by the same value.

- Record list entries

  Also known as Modified Lock List. DB2 members of a data sharing group store run-time information about modified locks owned by each member in the RLE. The information contained in the RLE is used to create retained locks when DB2 member abends. So if a lock is to be preserved across a subsystem abend, it must be stored in the RLE. The maximum number of modify locks related to database update processing that can be concurrently held and propagated to the CF is limited by the size of the RLE section. When 90% of the space allocated for the RLE is used, new modify lock requests will abnormally terminate with -904. As in the IRLM storage exhaustion, long and disruptive backouts would follow.

For the CF lock structure, sizing considerations are more complicated.

The actual number of modify locks that can be held in a given RLE section size is dependent on the CF level and the level of OS/390. (For more details, see *CFLEVEL considerations* at http://www.s390.ibm.com/pso/cftable.html.)

Starting with z/OS 1.2, IBM has introduced a new z/OS algorithm to decide whether to convert synchronous requests to asynchronous requests. This new z/OS function monitors CF service times for all (CACHE, LIST, and LOCK) structures and, based on new internal thresholds, determines which requests would be more efficiently executed asynchronously. Thresholds are different for simplex and duplex requests and for lock and non-lock requests.(For more details, review Washington System Center Flash # 10159.)

### 2.2.5 DB2PM statistics batch report

DB2PM reports come usually with two layouts, short and long, at a member or group level.

We should first concentrate on the long version of the report. When we are familiar with its capabilities, we can then switch to the short version.

The member level report header displays information such as Location, Group name, Member ID, Subsystem ID and time interval.

In section HIGHLIGHTS, besides information about date, time interval, and which member ID is being reported on, the important displays are the number of GETPAGES and Synchronous pages read.

The EDM POOL section presents detailed information about EDM POOL utilization. It is important here to make sure that there is at least 20% available space to accommodate unforeseen workload growth.

The LOG ACTIVITY section helps you monitor how well your logging activity is performing. You should pay special attention to the UNAVAILABLE OUTPUT LOG BUFF field. If you find any number other than zeros, it may be time to increase OUTBUFF in your zparm member.

The DYNAMIC SQL STMT section gives you the Global and Local Cache hit ratios.

DBM1 STORAGE STATISTICS (MB) section helps you monitor DBM1 virtual storage consumption.

The sections on member buffer pool information assist you in tuning and monitoring your buffer pool configuration.

The next sections give you information about group buffer pools at a member level.

At the Group level, DB2PM headers display Location, Group name, and the interval.

If you are in z/OS 1.2 and above, you may have to disregard the DATA SHARING LOCKING section. DB2PM is wrongly inflating false contention rate and they have announced dropping support for this item. You may find false contention rate information accurately displayed in RMF.

The section about group buffer pools provides the necessary information on a group level to tune your group buffer pool configuration.

These reports can be quite lengthy, but DB2PM is flexible enough to allow customization and deletion of unnecessary fields or a whole section. It also enables you to change the order in which sections and fields are presented.

To customize a report you may need to copy member DGOJINIT from the *.SDGOSAMP library into your personal JCL library.

Next, customize the DGOJINIT member by locating the high_level_qualifier field and updating it with your local DB2PM high-level qualifier.

You must allocate a PDS library with LRECL=80, BLKSIZE-6400.

Then, go to TSO option 6 and issue the EX MY.DSN(DGOJINIT) command. This opens a window with six options, which represent the various reports that can be customized.

Place the recently created PDS library in the DPMPARMS dataset field. Now you can start changing the DB2PM reports layout.

The new layouts will be saved in the library specified in the previous step. You must insert it into the DB2PM execution JCL in ddname DPMPARMS.

By customizing DB2PM reports, you can have a better presentation as well as save space in the JES output queues.

## 2.2.6 Table space partitioning

In this section we discuss why you may want to use table space partitioning in your PeopleSoft data sharing system. Detailed information about partitioning table spaces can be found the *DB2 UDB for z/OS Administration Guide* for your DB2 release.

### Partitioning to improve table space management

In both non-data sharing and data sharing environments, you may have to partition large table spaces due to their size and predicted growth. In both of these environments, partitioning a table space makes it manageable for housekeeping utilities such as Reorg, Runstats, and Image Copy.

## Partitioning to avoid I/O contention

In addition to the data management and housekeeping issues already mentioned, one of the main driving forces to implement partitioning is to avoid I/O contention for heavily used tables. This is achieved by partitioning the data and placing it on separate I/O paths and volumes.

You can choose a partitioning key that spreads your data evenly around the partitions in a linear manner. This type of partitioning often uses a random artificial key column.You can then benefit from reduced I/O contention on the partitioned tables.

When choosing this type of linear partitioning key, make sure you create enough partitions to hold the predicted growth in data without the partitions becoming too big. You should also consult your application developers to ask whether there is any benefit from clustering the data in any particular ascending or descending order using another column in the key. Note that prior to DB2 V8 the partitioning key also had to be the clustering key.

When choosing linear partitioning, reducing your I/O contention by spreading your data should be beneficial. However, you may not be able to control member affinity for the batch jobs or transactions accessing the data. This may result in cross-member interest in the group buffer pools.

A second method of partitioning is to choose a partitioning key that results in each partition being filled in turn as the data is created. With this type of partitioning, you may be able to parameterize your jobs to operate separately on each partition. This is because the key may be based on an account number, a customer number, or a processing period.

When you are considering the separation of partitions to different I/O paths and volumes you may ask yourself why this is necessary if you have the latest disk hardware. The reason this is still important is that OS/390 and z/OS still control access to the volume at the UCB level. There will be queueing and contention at the volume UCB. If you have the Parallel Access Volume enhancement (PAV) implemented, then this contention will be reduced significantly.

## Improving I/O performance for nonpartitioning indexes

If your partitioned tablespace has also a nonpartitioning index or indexes then these could be an I/O bottleneck if you have concurrent PeopleSoft processes making lots of insertions, deletions, or updates into the table.

You may see high insert and update times in the PeopleSoft single record performance statistics. To reduce the insert, update, and delete times for your applications you should consider using the PIECESIZE option for these nonpartitioning indexes. This is especially relevant to nonpartitioning indexes that

support large tables, but can help performance on any partitioned or nonpartitioned table. Figure 2-7 illustrates how a nonpartitioned index can be a performance bottleneck.

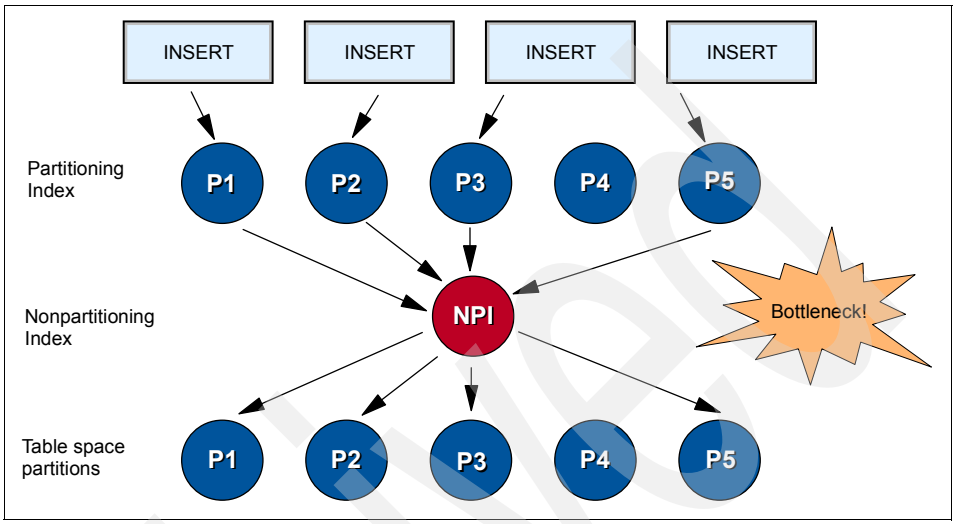



*Figure 2-7 Nonpartitioning index bottleneck without PIECESIZE option*

By using the PIECESIZE option you can split your nonpartitioning indexes into several physical datasets. These datasets can then be placed on separate volumes and I/O paths to reduce the I/O contention and help balance the I/O across the datasets of the index. This is illustrated in Figure 2-8 on page 47.

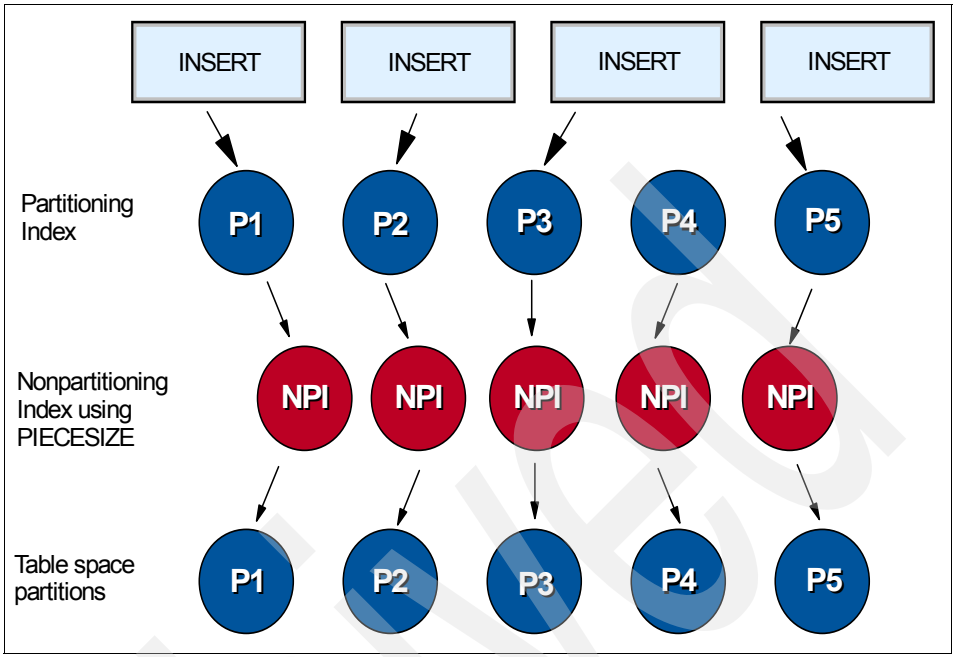



*Figure 2-8 Non partitioning index with PIECESIZE option*

For further considerations about choosing a PIECESIZE, consult the DB2 SQL reference manual for your DB2 release.

## 2.2.7 Managing DBM1 virtual storage

For many years now, the 31-bit addressability of MVS™ and then OS/390 has required various techniques for tuning around the fact that only 2 GB (in fact, somewhat less than this) is available to an address space. This has led to features such as Dataspace and Hiperspace™, which are exploited by DB2 for OS/390 and z/OS as a means of mitigating the effect of the 2 GB addressability limit. In this chapter we discuss the use of these facilities specific to a PeopleSoft environment.

There are three major contributing environmental factors:

- How much central (real) storage is available in the LPAR
- How many DB2 subsystems are run in this LPAR
- What other applications will require memory resource in this LPAR

The overriding factor in storage (real and virtual) management is the initial condition of how much storage is allocated to the LPAR in consideration. Obviously where multiple LPARs are involved, there is also a consideration of

how much storage to allocate to each LPAR from the physical memory available on the S/390® or z/Series server. This allocation choice is somewhat beyond the scope of this book, but the general philosophy that follows (that of making the optimum use of available resources by moving resources to the source of greatest benefit) also applies to the question of LPAR storage.

Another issue to be considered is how much storage will be consumed by non-DB2 tasks in this LPAR. It is widely recommended by PeopleSoft and IBM that production PeopleSoft systems should reside in an LPAR that contains only production, as opposed to development, testing, or other activities. It is possible that some of these other tasks will be unrelated to the PeopleSoft system's DB2 subsystem; that is, there may be other non-PeopleSoft production work running in the LPAR. The storage requirements of such tasks are beyond the scope of this book. If the requirements are known, they can be effectively subtracted from the available storage to be allocated. For non-production systems, there are several possibilities for number of LPARs and how many and what types of PeopleSoft systems will be run in each LPAR.

System tasks will also consume a significant amount of memory resources. The requirement for real storage, as opposed to virtual storage, will depend on how much memory the system tasks need, which is not paged to auxiliary storage due to inactivity of the pages.

Finally, after taking into account all other possible consumers of resource in the LPAR, we have an idea of how much storage we can allocate to the OS/390 or z/OS resident components of a PeopleSoft system. These components are DB2 subsystems and possibly OS/390 application servers. In the case of running multiple DB2 subsystems for different PeopleSoft systems, the allocation of memory between them will be a factor of the workload to be run within the subsystems and the performance levels expected in the systems.

Therefore, the process of tuning starts from an understanding of the available resources in the system in question. Second you must understand the concept of virtual storage constraint, which ultimately can lead to subsystem failure through getmain failures, as opposed to real storage constraint, which can lead to performance problems through paging and auxiliary storage shortages. The process of tuning that follows these understandings can then take place in a structured manner, leading to a proposed first configuration, and then subsequent refinement of this configuration through controlled changes and measurement. With a knowledge of how much real storage DB2-related tasks are able to occupy, we can then decide which of the available virtual storage constraint relief facilities it is appropriate to use.

## Managing the DSC size

With the advent of the EDM Pool dataspace available from DB2 V6 and subsequent releases, the process of managing DSC resources has been greatly simplified. The EDM Pool dataspace is enabled through DSNZPARM EDMDSPAC, and it defaults to 40 MB if an EDMPOOL size greater than 40 MB is specified. By allocating DSC pages in a dataspace, the allocated pages no longer have the detrimental effect on the DBM1 address space of consuming a large number of EDM Pool pages to achieve adequate Dynamic SQL performance.

As PeopleSoft relies heavily on the performance of Dynamic SQL, effective caching is extremely important, as it reduces the number of times DB2 has to consume the resources required to perform a full prepare. If an EDM Pool dataspace is utilized, which is highly recommended, then it is also recommended to make the EDM Pool dataspace large enough to contain all prepared Dynamic SQL statements. This is because it is probably more resource-effective to page in a DSC page related to the dataspace from auxiliary storage than it is to prepare the statement again.

An approach to finding an appropriate value for EDMDSPAC is to initially set it high, for example 300 to 400 MB, and periodically monitor the number of pages used for DSC during a time of normal activity (for example, taking daily snapshots). The number of pages should eventually stabilize at a peak value, and subsequently reducing EDMDSPAC to 5-10% more than this will result in an appropriate level.

The most appropriate time to check the number of DSC pages used is immediately prior to shutting down the PeopleSoft system following a significant period of uptime. It should be noted that within the statement cache, SQL statements exist that probably will not be executed again prior to DB2 stopping, such as initial reads of PeopleSoft buffered tables. However, things like application server restarts may result in them being run again, and the minimal cost of having them paged out to auxiliary storage is considered a small price to pay for possibly avoiding DB2 prepares.

By having all SQL statements in DSC, there are also some benefits to be gained by IFCID 318: Monitoring of the DSC statement performance is enabled, as any given statement can subsequently be located in the cache.

## Managing open datasets

The number of open datasets in a given PeopleSoft system is somewhat outside the control of a DB2 System Administrator or PeopleSoft Administrator, and it is primarily a result of the PeopleSoft modules or components employed in the PeopleSoft system. Generally the target is to enable DB2 to open the required

datasets through normal PeopleSoft-imposed activity, that being to satisfy the flow of SQL requests arriving in the form of Dynamic SQL.

Periodically, some other activity may occur in the system that results in other datasets being opened that are not usually used in the normal course of PeopleSoft activity. These can include:

- DB2 utilities such as RUNSTATS being performed, which again may be run against tablespaces that normally remain closed, as the Administrator is performing a complete job by updating statistics on all tables.

### 2.2.8 Configuring DB2 sort pool and workfiles

PeopleSoft applications can perform many sorts in DB2. The speed of these sorts is a very important factor in the performance of the application. Therefore you should take care to configure your DB2 sort pool and work file table spaces used for sorting for maximum efficiency.

You should configure your zparms SRTPOOL to make your DB2 sort pool the maximum size allowed by your DB2 release. However, remember that this sort pool is not a shared memory area in the DBM1 address space. Each DB2 thread will allocate a sort pool of this size if it requires it. So this may have a significant impact on your DBM1 virtual storage.

Next, begin by deciding how many workfile table space datasets you can afford to allocate, bearing in mind that each workfile table space dataset should be placed on its own disk volume and I/O path to avoid I/O contention. We recommend a minimum of five workfile table space datasets for each data sharing member. As a starting point, these table spaces should be sized at 2 GB and each table space should be the same size. You should place these table spaces on shared disks, because if a query uses sysplex parallelism, the controlling member of the data sharing group may need to read from these table spaces. Placing the table spaces on shared disks also keeps a data sharing member connected to its work files if it fails over to another LPAR.

When you have decided how many workfile table spaces you will define, you must then place these table spaces into their own local buffer pool. This buffer pool should not be backed by a group buffer pool. For best use of the sort datasets, the buffer pool should be sized as large as you can make it within your DBM1 virtual storage constraints. Otherwise, it will be a limiting factor in how many sort datasets DB2 can use.

If you are not using a hiperpool when making the buffer pool settings, you should set the DWQT and VDWQT thresholds to 90% to avoid writing to the DASD too early. However, if you set these thresholds to these high values, you run the risk of flooding the I/O subsystem with large amounts of pages to be written when the

threshold is reached. If you see this happening, then consider leaving the thresholds at their lower default values so that you do not write large amounts of pages to the disk when they are reached.

If you are using a hiperpool, you should set the DWQT and VDWQT thresholds low to force the pages asynchronously out to disk. After the pages have been written to disk they will be marked clean, and therefore be eligible for migration to the hiperpool. When they are in the hiperpool they can be brought in during the merge phase of a sort without any I/O.

If I/O does occur in the sort process then DB2 will use sequential prefetch to bring the pages back into the buffer pool. So if the buffer pool is constrained for sequential prefetch, then sequential prefetch will be disabled and performance will suffer. You should monitor your work file buffer pools to ensure that sequential prefetch is not being disabled.

You also should monitor your sort buffer pool activity to see if the performance-inhibiting thresholds are being reached. Keep in mind that processing activity in these buffer pools is almost all sequential.

If you are doing large sorts, and after tuning the buffer pool you are still seeing significant I/O activity on your workfile volumes, then you should consider allocating this buffer pool to a data space. This is possible if you are at DB2 Version 6 or later and you are using a 64-bit zSeries machine and OS/390 2.10 or z/OS. Before you use data spaces, ensure that you have enough central storage to hold them. If you cannot hold the dataspace in central storage then their use is not recommended. If you are not using a 64-bit zSeries machine, you should consider allocating the buffer pool to a hiperpool.

The efficiency of sorts can be analyzed in the DB2PM SQL activity reports. The sort summary trace record IFCID 0096 can be used to analyze your sorts and is included in performance trace class 3.

For more criteria on which to base the number and size of workfile table spaces, consult the DB2 Administration Guide for your DB2 release.

3

# DB2 Connect architectures for PeopleSoft

This chapter describes the DB2 Connect database connections and how DB2 Connect performs load balancing.

We describe the following topics:

- PeopleSoft connectivity architecture
- DB2 Connect EE configurations with PeopleSoft
- DB2 Connect EE thread pooling techniques
- Networking considerations - VIPA

## 3.1 PeopleSoft connectivity architecture

Figure 3-1 shows PeopleSoft three-tier architecture with database server on DB2 for z/OS. PeopleSoft uses DB2 Connect EE to connect to its database on DB2 for z/OS.

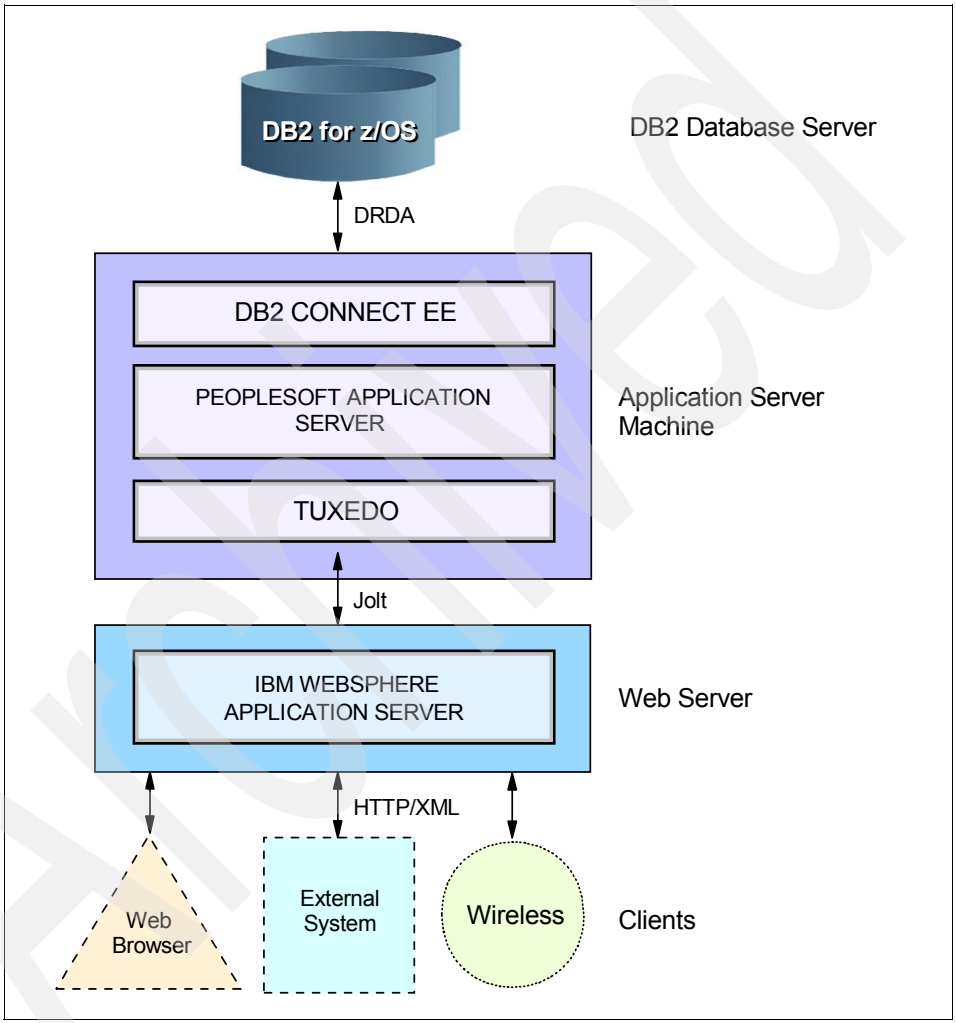



*Figure 3-1 PeopleSoft three-tier architecture with database server on DB2 for z/OS*

## 3.2 DB2 Connect EE configurations with PeopleSoft

DB2 Connect EE can be configured in different ways for PeopleSoft:

- Direct connection configuration
- Connection server configuration

### 3.2.1 Direct connection configuration

There is a direct connection between the application server and the database server. DB2 Connect EE is on the same machine as the PeopleSoft application server as shown in Figure 3-2.

The DB2 Connect EE enables the application server to connect to the DB2 database server on z/OS.

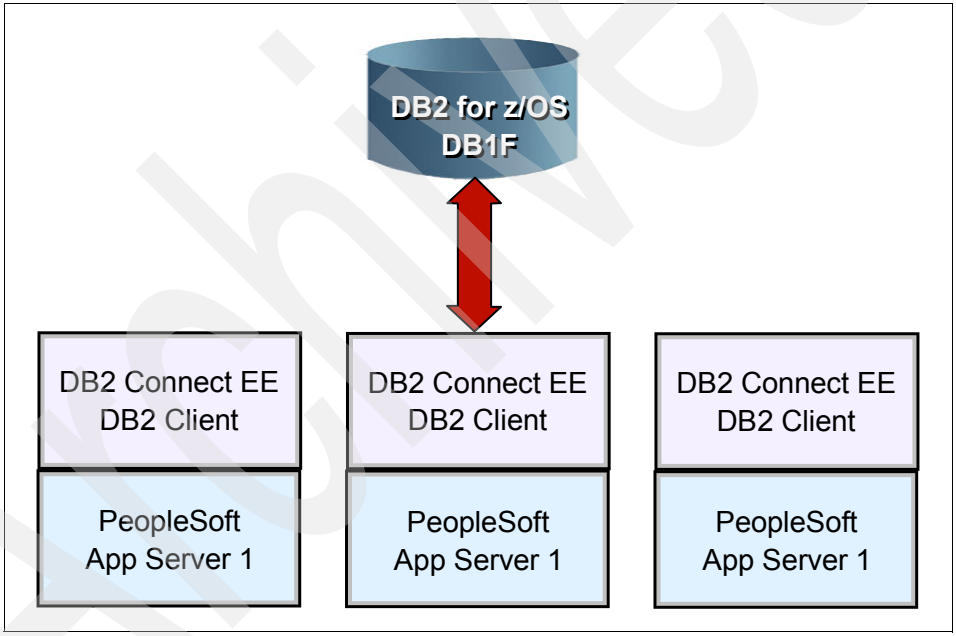



*Figure 3-2 Direct connection between the application server and the database server*

With this direct configuration, you must:

- Have a DB2 Connect EE installed on each application server machine.
- Catalog the DB2 location name (DB1F) at each DB2 Connect EE server.
- Create and configure an application domain on each PeopleSoft application server.

### 3.2.2 Connection server configuration

In this scenario DB2 Connect EE acts as a connection server. DB2 Client is installed on each application server machine (Figure 3-3).

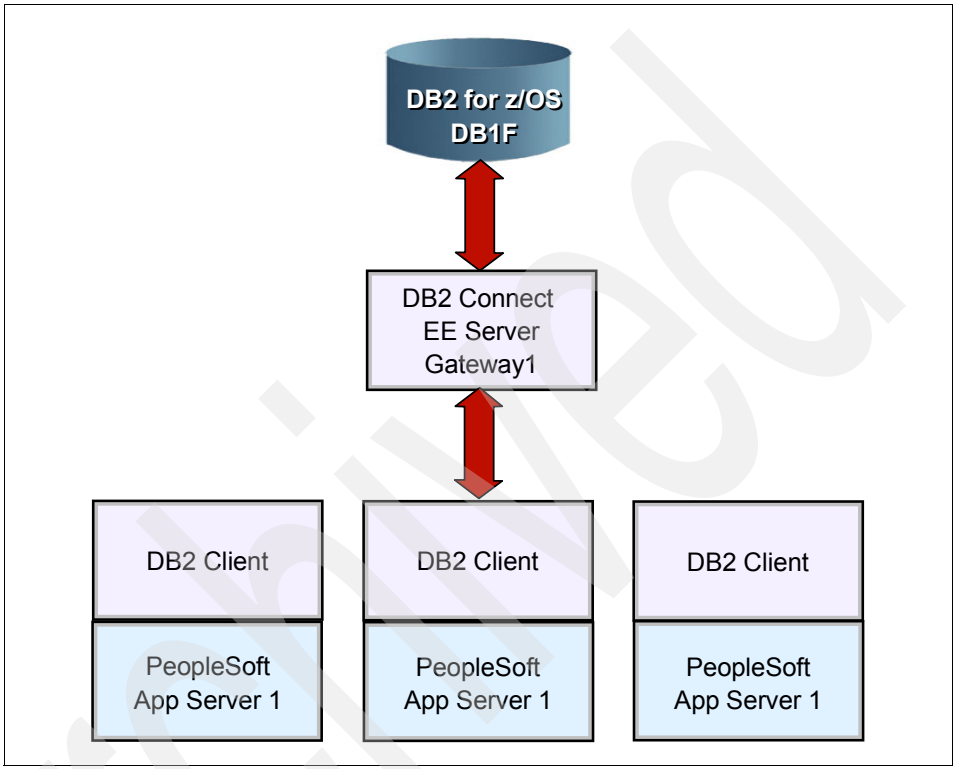



*Figure 3-3 Gateway connection between the application server and the database server*

For this setup, you must:

- Catalog the database location name DB1F at the DB2 Connect EE connection server.
- Catalog the DB2 Connect EE connection server at each DB2 Client.
- Create and configure an application server domain on each PeopleSoft application server.

## 3.3 DB2 Connect EE thread pooling techniques

Establishing a connection from a DB2 Connect server to the host requires computing resources and time. In an environment where thousands of client applications frequently connect to and disconnect from the host through the DB2 Connect server, a substantial part of processing time is spent establishing connections and dropping connections. This is especially evident in Web environments where each visit to a Web page can require building a new connection to the database server, performing a query, and terminating a connection.

To reduce this overhead, DB2 Connect EE uses thread pooling techniques to reduce resources required on the zSeries host database servers. It accomplishes this goal by concentrating the workload from all applications into a much smaller number of host database server connections.

We describe the following thread pooling techniques:

- Connection pooling (default)
- Connection concentrator

Both connection pooling and connection concentrator are described in the *IBM DB2 Connect User's Guide, Version 8,* SC09-4835.

### 3.3.1 Connection pooling

Connection pooling is a technique that enables reuse of an established connection infrastructure for subsequent connections. Connection pooling is activated by default. When a DB2 Connect instance is started, a pool of coordinating agents is created. When a connection request comes in, an agent is assigned to this request. The agent connects to the DB2 server and a thread is created in DB2. When the application issues a *disconnect* request, the agent does not pass this request along to the DB2 server. Instead, the agent is put back into the pool. The agent in the pool still owns its connection to the DB2 server and the corresponding DB2 thread. When another application issues a connect request, this agent is assigned to this new application.

Figure 3-4 on page 58 shows the DB2 Connect EE connection pooling technique.

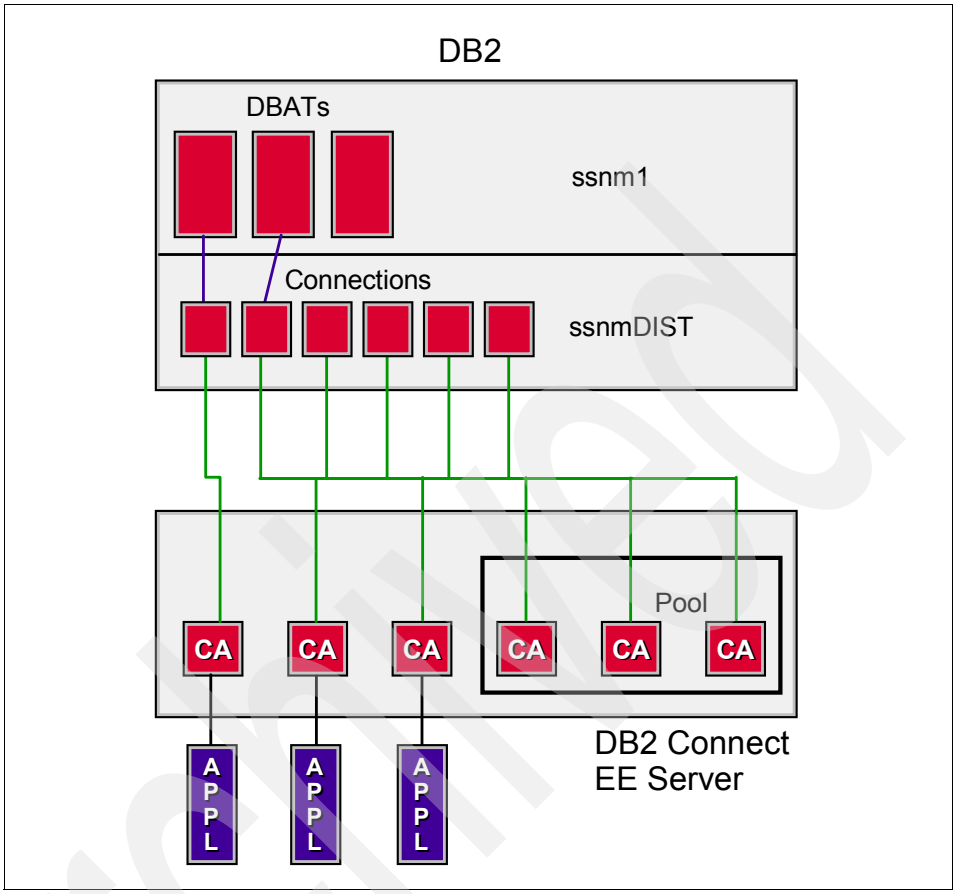



*Figure 3-4 DB2 Connect EE connection pooling technique*

Connection pooling is transparent to applications connecting to the host through DB2 Connect. When an application requests disconnection from the host, DB2 Connect drops the inbound connection with the application but keeps the outbound connection to the host in a pool. When a new application requests a connection, DB2 Connect uses one from the existing pool. Using the already-present connection reduces the overall connection time, as well as the high CPU connect cost on the host.

DB2 Connect coordinating agents (CA) can be in one of two states: idle or active. An agent is active when it is executing work for an application. When this work is completed the agent goes into an idle state awaiting further work from the same or a different application.

All idle agents are kept together in what is known as the idle agent pool. You can configure the size of this pool using the NUM_POOLAGENTS configuration parameter. This parameter equals the maximum number of idle agents you wish the system to maintain. Setting this parameter to zero is equivalent to turning off the connection pooling feature.

DB2 Connect does not establish connections to the database before receiving its first client request. If you wish, however, you may fill the pool of idle agents before any clients make a request. The pool can be filled on startup using the NUM_INITAGENTS configuration parameter. This parameter determines how many idle agents should be created at start-up time. These idle agents will not initially have connections to the host database server.

When a client requests a connection to the host, DB2 Connect attempts to get an agent from among those in the pool that have a connection to the host database server. If that fails, it will try to find an available agent in the idle pool. If the pool is empty, DB2 Connect will create a new agent.

You can control the maximum number of agents that can be concurrently active using the MAX_COORDAGENTS configuration parameter. When this number is exceeded, new connections will fail with error sqlcode SQL1226. (This code means that the maximum number of concurrent outbound connections has been exceeded.)

The db2 registry variable DB2CONNECT_IN_APP_PROCESS enables applications running on the *same machine* as DB2 Connect EE either to have DB2 Connect run within the applications process, default behavior, or to have the application connect to the DB2 Connect EE Server and then have the host connection run within an agent. For an application to use connection pooling, the connections to the host must be made from within the DB2 Connect EE Server agents. Thus DB2CONNECT_IN_APP_PROCESS must be set to NO.

In summary, the differentiators for connection pooling are:

- Connection pooling is *on* by default.
- There is a 1-1-1 relationship between applications, coordinating agents, and connections into DB2.
- The coordinating agent and corresponding host connection are returned to the pool at *disconnect* time.

However, connection pooling has a minor drawback when used with another capability of DB2 Connect EE: connection to a DB2 data sharing group (Parallel Sysplex awareness). With Parallel Sysplex awareness activated in DB2 Connect EE, DB2 Connect EE processes the information about the availability of the members in a DB2 data sharing group only on the creation of a new connection to DB2 UDB for OS/390. With connection pooling also activated, there is a

chance that connections would remain with a particular member of the data sharing group even if that member had problems. DB2 Connect EE would also not use sysplex information to determine which of its pooled connections would be the best connection to be reused for a client request. Finally, if an outage occurred on a member of the data sharing group, all client connections with that member would receive connection failures regardless of whether the clients were active in the database server.

Thus, the need arose to have DB2 Connect EE provide a way to share a server task between many client connections without having to wait for a particular client to disconnect and, using sysplex support, continually balance its connections to the members of the DB2 for z/OS data sharing group.

### 3.3.2 Connection concentrator

Connection concentrator builds on the features of connection pooling by providing improved load balancing in Parallel Sysplex configurations. Connection concentrator technique has a more sophisticated approach to reducing resource consumption for very high volume Online Transaction Processing (OLTP) applications. This function can dramatically increase the scalability of your DB2 for z/OS and DB2 Connect solution while also providing for transaction-level load balancing in DB2 for z/OS data sharing environments.

With connection pooling, one application has to disconnect before another one can reuse a pooled connection. Rather than having the connection become free for use by another client at client disconnect, connection concentrator enables reuse of a server task when an application performs a *commit* or *rollback.*

Figure 3-5 on page 61 shows DB2 Connect EE connection concentrator technique.

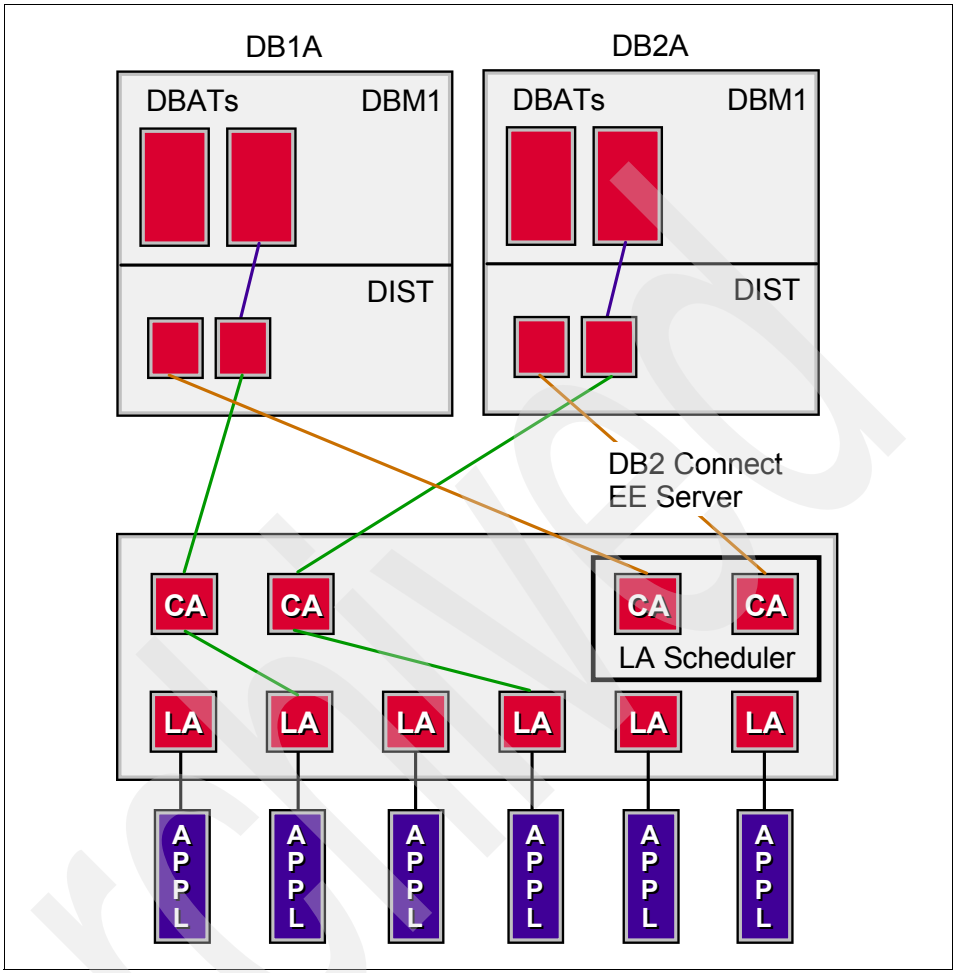



*Figure 3-5 DB2 Connect EE connection concentrator technique*

In this architecture, there is a many-to-one relationship between coordinating agents and host connections. That is, the relationship of coordinating agents (X) to host connections (Y) to is now X >= Y.

Connection concentrator introduces a concept of Logical Agent (LA), which handles user context, while coordinating agent (CA) continues to own DB2 connection and thread. A new application user is assigned an LA upon connection. CA is needed to pass SQL to DB2 so one is assigned as soon as new transaction is initiated. The key to this architecture is the fact that CA is disassociated from the LA and is returned to the pool when transaction completes (commit/rollback). Another key feature is the method of assigning CAs to new transactions in a data sharing environment. DB2 Connect implements a

sophisticated scheduling algorithm that uses z/OS Workload Manager (WLM) information to distribute workload across members of a data sharing group according to criteria set up in WLM. WLM is not only aware of the load on each member but also their availability. This enables DB2 Connect to transparently relocate work away from failed or overloaded members to those that are up and underutilized. DB2 Connect connection concentrator is activated when you set the number of maximum logical agents higher than the number of coordinating agents.

Certain conditions have to be met when a client application reaches a commit point so that the real agent can be considered reusable by other logical agents. Those conditions are:

- Open WITH HOLD cursors must not exist on the connection. Transactions that do not close WITH HOLD cursors will still go through, but will be assigned a dedicated worker agent and hence will not be able to use the concentrator's full feature set.
- Declared global temporary tables that have the ON COMMIT PRESERVE ROWS option do not exist.
- Packages with the KEEPDYNAMIC YES bind option have not been referenced.

DB2 Connect EE can keep track of the state of cursors used by any client. When a commit occurs and the cursor WITH HOLD does not exist, DB2 Connect EE knows that it can break the tie between the logical and real agent. However, DB2 Connect EE cannot keep track of the state of the latter two conditions described above. In DB2 for z/OS V7, support was added to inform any DRDA® requester (of which DB2 Connect EE is one) that a connection could be released. Also, DB2 for z/OS V7 sends back to the DRDA requester the current values of the special registers (for example, CURRENT SQLID) on the response to a commit request. DB2 Connect EE must be upgraded to *at least* Fixpak 2 to make use of this support in DB2 for z/OS V7.

When DB2 Connect EE is also accessing DB2 data sharing groups in connection concentrator mode, it frequently gets the status of the group's members. DB2 Connect EE uses this information to continually balance the number of connections it has to each member of the group and determine to which member a client's next transaction should be routed. The best feature of this support is what happens if an outage occurs on one of the members in the DB2 data sharing group: Only those client connections that are actually processing transactions in the particular member of the data sharing group receive connection failures. All other clients remain connected to the DB2 Connect EE server or continue their transactions at other members of the DB2 data sharing group.

Connection Concentrator improves performance for long-running connections that require load balancing where an application server provides connections for many clients. For best performance, it is suggested that there be enough agents available in the connection pool to contain all active clients. This avoids unnecessary connection reuse outside of the load balancing process.

### 3.3.3 Connection pooling versus connection concentrator

Connection pooling and connection concentrator seem to have similarities, but they differ in their implementation and they address different issues. Connection pooling helps reduce the overhead of database connections and handle connection volume. Connection concentrator helps increase the scalability of your DB2 for z/OS and DB2 Connect solution by optimizing the use of your host database servers.

When using connection pooling, the connection is only available for reuse after the application owning the connection issues a disconnect request. In many two-tier client-server applications, users do not disconnect for the duration of the workday. Likewise, most application servers in multi-tier applications establish database connections at server start-up time and do not release these connections until the application server is shut down.

In these environments, connection pooling has little if any benefit. However, in Web and client-server environments where the frequency of connections and disconnections is higher, connection pooling produces significant performance benefits. The connection concentrator allocates host database resources only for the duration of an SQL transaction while keeping user applications active. This allows for configurations in which the number of DB2 threads and the resources they consume can be much smaller than if every application connection had its own thread.

When it comes to fail-safe operation and load balancing of workload, connection concentrator is clearly the right choice as it allows reallocation of work with every new transaction. Connection pooling, on the other hand, can only offer very limited balancing and only at connect time.

### 3.3.4 Sysplex-aware connections

When DB2 Connect EE is sysplex-aware, requests can be routed to any available DB2 data sharing group member. The most suitable member gets the connection.

With sysplex awareness, DB2 Connect EE receives information from both DB2 and Workload Manager to enable connection load balancing between the

members of the data sharing group. DB2 Connect EE receives this information at the creation of each *new* connection.

DB2 Connect has the following configuration requirements for sysplex exploitation:

- On a DB2 Connect EE server, sysplex exploitation is enabled by default. (It can also be turned off by setting the DB2SYSPLEX_SERVER profile variable to the value zero).
- Sysplex exploitation will not be used for a given database unless the DCS directory entry for that database contains `sysplex` (not case-sensitive) in the 6th positional parameter.
- The DB2 registry variable DB2CONNECT_IN_APP_PROCESS can be used to enable clients running on the same machine as the DB2 Connect EE server to exploit sysplex support. If you use a gateway server to connect to a database, then setting DB2CONNECT_IN_APP_PROCESS to NO is not required.
- DB2 Connect EE connection concentrator feature must be enabled to exploit load balancing at transaction boundaries (commit/rollback).

## 3.4 Networking considerations: VIPA

Virtual IP addresses (VIPAs) enable additional high availability at the networking level.

VIPAs are IP addresses that are independent of any particular network interface. To an external router, VIPA appears simply as an address or subnet that is reachable via the hosting stack. The benefit of using VIPA as an application address to a z/OS TCP/IP with multiple physical links is that a failure of any one link will not disrupt connectivity to the application. As long as there is a network path from the remote client to the TCP/IP hosting the VIPA and the server application, the client and server application interact uninterrupted. A VIPA provides independence from any particular adapter, but is still for the most part tied to a particular z/OS TCP/IP stack.

VIPAs are not associated with any physical link, so there is no particular reason that a VIPA should be associated with one and only one TCP/IP in a sysplex. With dynamic VIPA, if one of the members in the sysplex suffers an outage, one of the remaining members can become the owner of the virtual IP address, providing a level of uninterrupted service.

When used in the PeopleSoft three-tier configuration with DB2 Connect EE, dynamic VIPA in a sysplex configuration ensures client connectivity, without

having to catalog each member and change the location name, as long as at least one member is available in the sysplex.

Because DB2 Connect EE handles the load balancing and can detect when a member in the sysplex fails, the dynamic VIPA mechanism participates only in the initial connection from the application server to the database server.

For further details about dynamic VIPAs, refer to the white paper *Dynamic VIPA routing and workload balancing in a DB2 data sharing environment.* You can find it at:

http://www-1.ibm.com/support/docview.wss?rs=64&uid=swg27000593

4

# DB2 Connect EE setup

This chapter describes the test environment and DB2 Connect setups done at the Silicon Valley Laboratory in order to test and understand how load balancing was achieved by the connection concentrator feature of DB2 Connect EE V8 for the PeopleSoft V8.43 applications.

We discuss the following topics:

- PeopleSoft test environment
- DB2 Connect EE configuration
- Application server domain configuration
- Validating sysplex awareness setup in DB2 Connect

## 4.1 PeopleSoft test environment

We have a three-tier configuration, as shown in Figure 4-1.

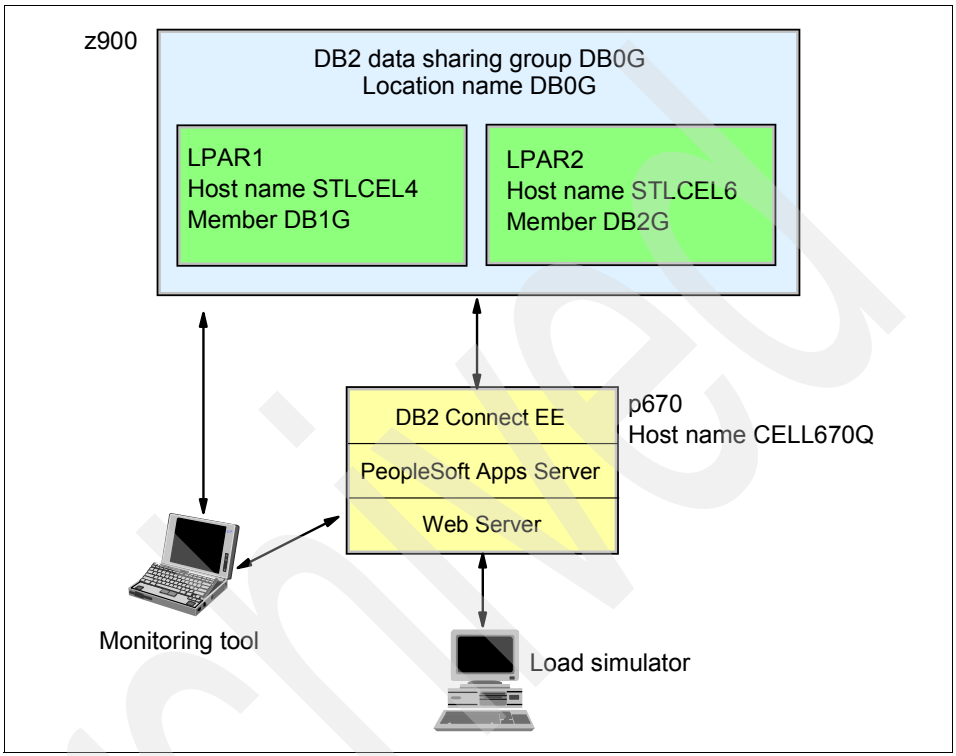



*Figure 4-1 PeopleSoft test environment*

The objective of the demo is to analyze the PeopleSoft workload balancing between the Application server and the two members of the database server in a three-tier configuration.

On the client tier, we use a Web load simulator, simulating 300 users. We also have a monitoring tool to observe and track the load balancing mechanism.

The middle tier has a Web server and one PeopleSoft application server processing users' requests and connecting to the database server via the DB2 Connect EE middleware.

The data tier has a Parallel Sysplex data sharing group with two members where the PeopleSoft-DB2 data server resides.

For demo purposes, we use the following system environment:

***z900 machine***

- z/OS V1.2 Parallel Sysplex
  - 2 LPARs: host names STLCEL4 and STLCEL6
  - 8 processors floating between both LPARs
  - 8 GB memory in each LPAR
- DB2 for z/OS V7 where the PeopleSoft data server resides

  Two-member data sharing environment, including:
  - One DB2 data sharing group DB0G
    - Member DB1G
    - Member DB2G
  - Host name STLCEL4 (or IP address 192.158.11.22)
  - Location name DB0G
- DB2 Performance Expert for z/OS V1.1 (server)

***p670 machine***

- AIX® V5.2

  1 LPAR: host name CELL670Q with 8 CP
- PeopleSoft V8.43.03 with Fixpack 3
- DB2 Connect EE V8 with Fixpack 2 and a beta APAR IY48963, which will be available in Fixpack 4
- WebSphere V4.and Fixpack 3
- DB2 Performance Expert for Multiplatform (server)
- DB2 Performance Expert Agent

***Client workstation***

- Windows® 2000
- Load simulator: MS Web application stress tool V1.1
- DB2 Performance Expert (client)

## 4.2 DB2 Connect EE configuration

We installed DB2 Connect EE V8 with Fixpak 2 and the beta APAR IY48963. We show the two ways of configuring DB2 Connect EE, by using either CCA or CLP.

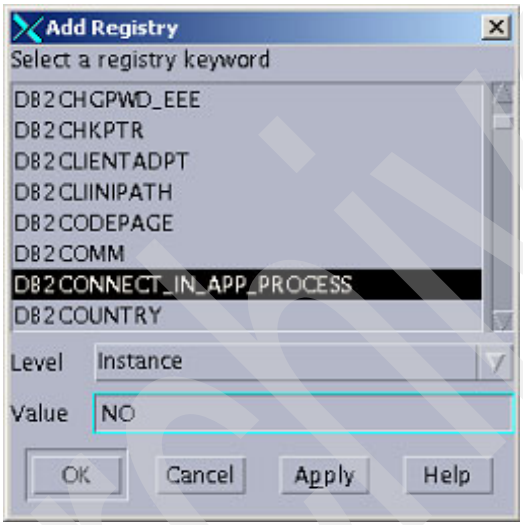
## 4.2.1 Using CCA

Launch DB2 Client Configuration Assistant (CCA) either from the Start menu or by issuing the **db2ca** command from DB2 CLP.

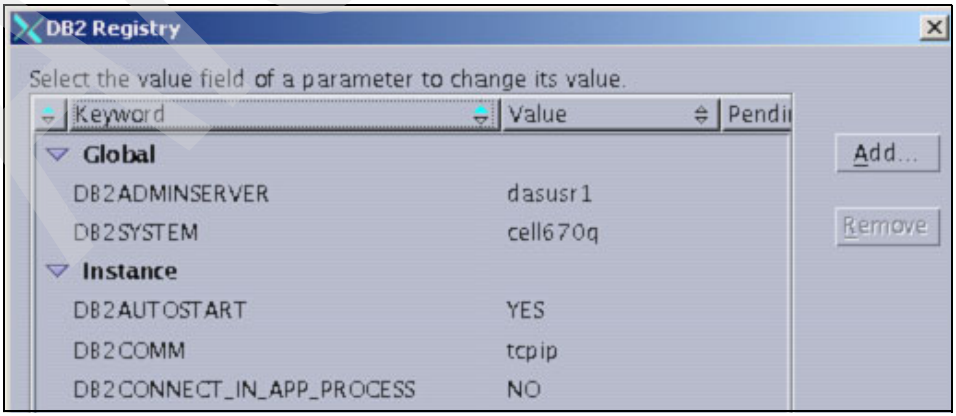
### Update DB2 registry

To set DB2CONNECT_IN_APP_PROCESS=NO:

1. Select **Configure** → **DB2 Registry**, and click **Add** to open the Add Registry window.
2. Highlight **DB2CONNECT_IN_APP_PROCESS**, keep Level as **Instance**, and set Value to NO. Click **OK** to update registry changes.



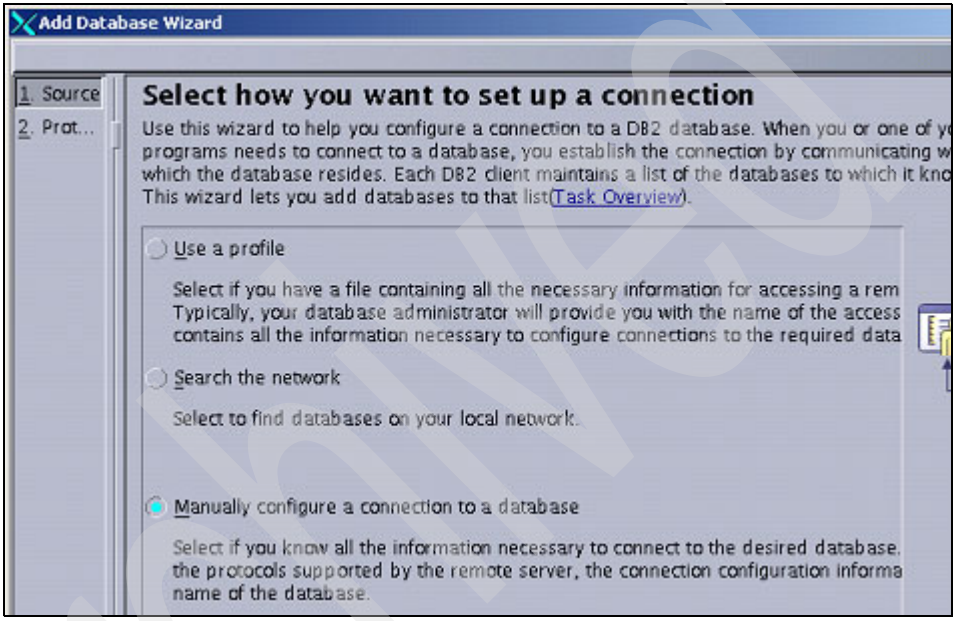

3. DB2CONNECT_IN_APP_PROCESS appears in the registry.



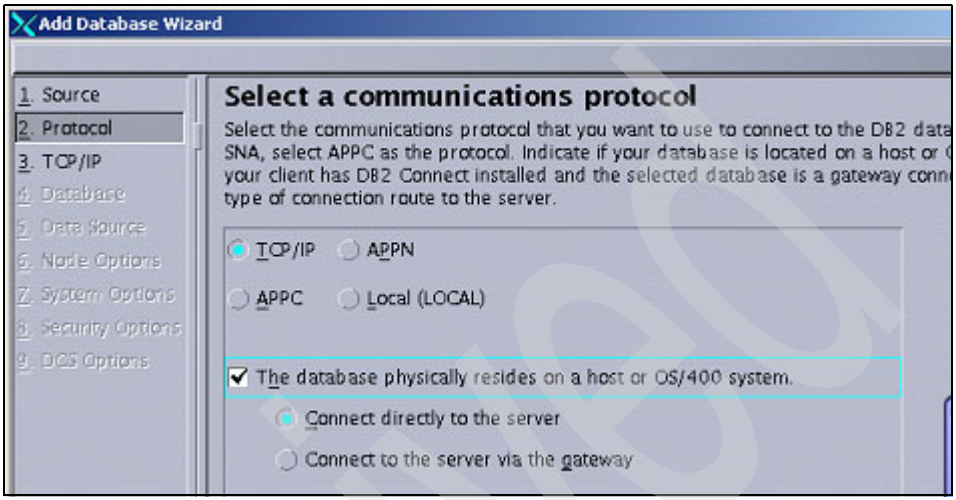
## Configure a connection to the database

Using DB2 Client Configuration Assistant (CCA), do the following:

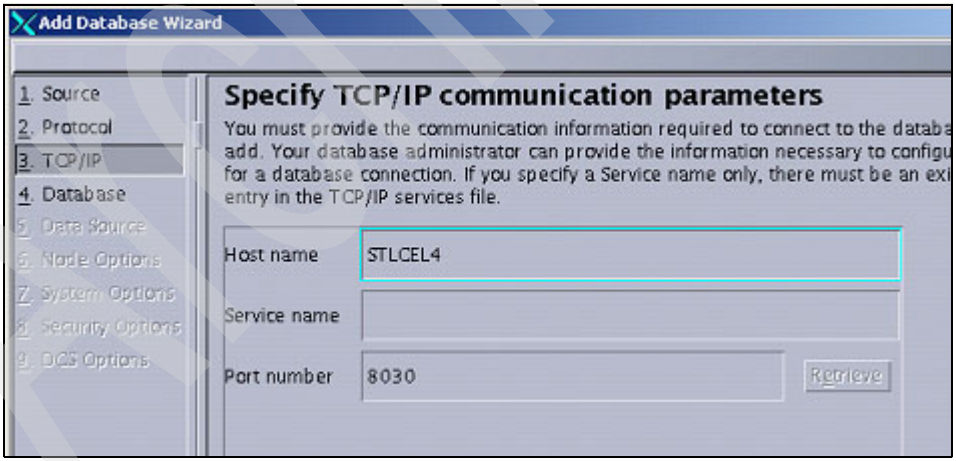
1. Launch DB2 Client Configuration Assistant either from the Start menu or by issuing the **db2ca** command from DB2 CLP.
2. From the menu, click **Selected** and choose **Add Database Using Wizard**.
3. Select **Manually configure a connection to a database**. Click **Next**.



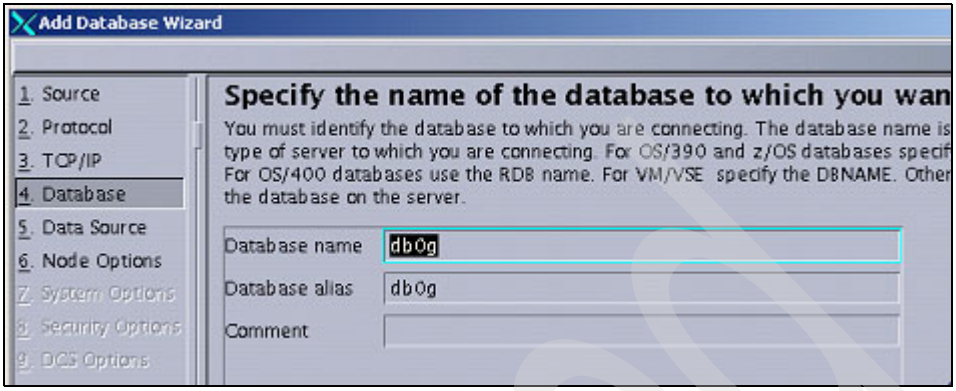
4. On the next window, choose **TCP/IP**.
5. Select **The database physically resides on a host or OS/400® system**, and choose **Connect directly to the server**. Click **Next**.



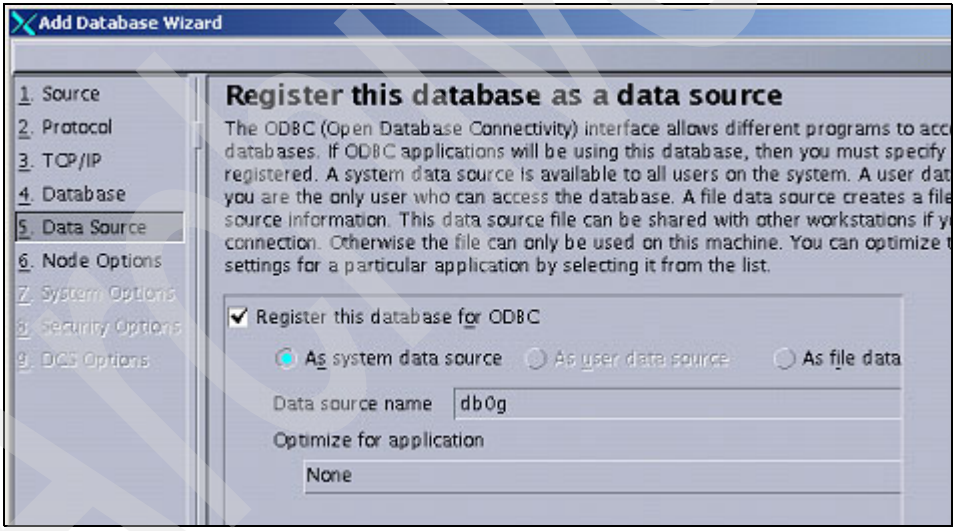

6. On the next window, enter `STLCEL4` for Host name and `8030` for Port number. Click **Next**.



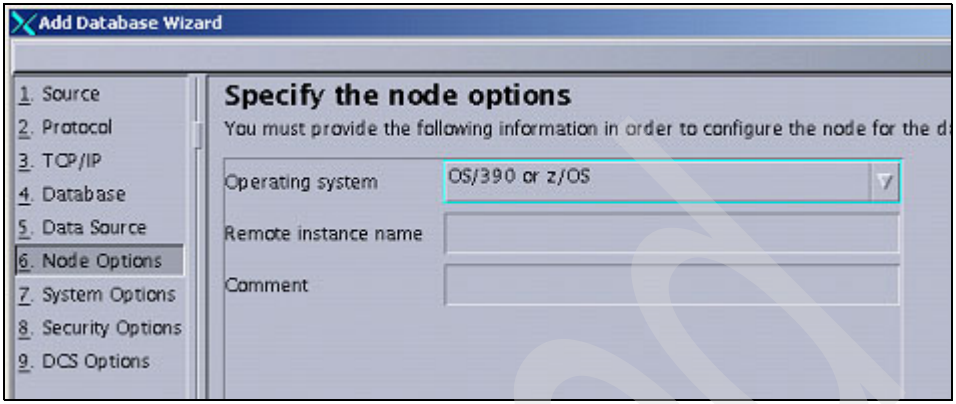
7. On the next window, enter `db0g` for the Database name. Click **Next**.



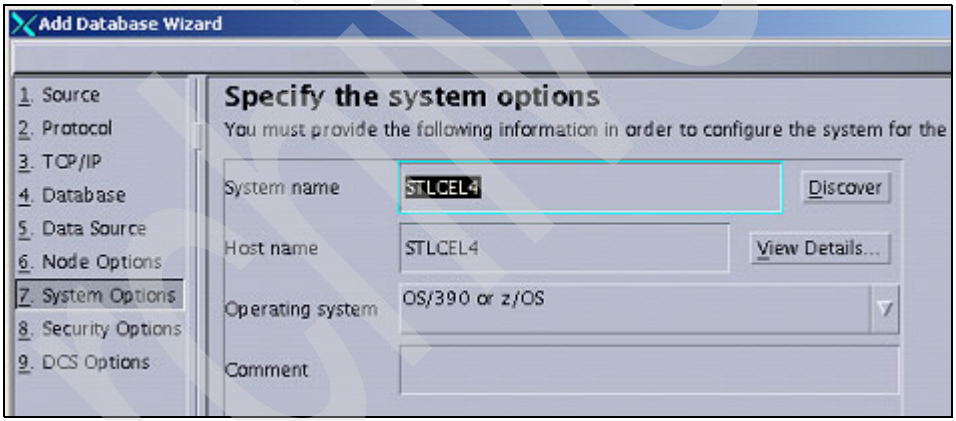

8. On the next window, select the check box for **Register this database for ODBC**, and select **As system data source**. Click **Next**.



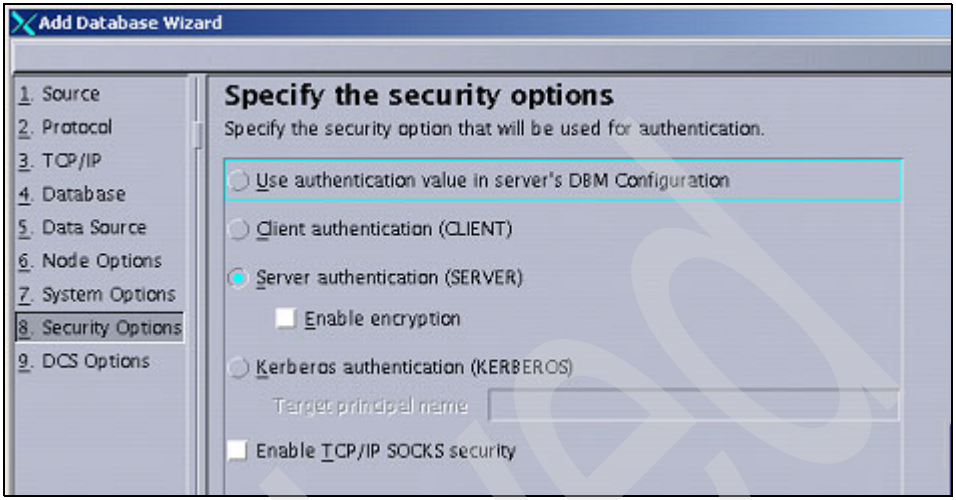
9. Choose **OS/390 or z/OS** from the pull-down menu, and click **Next**.



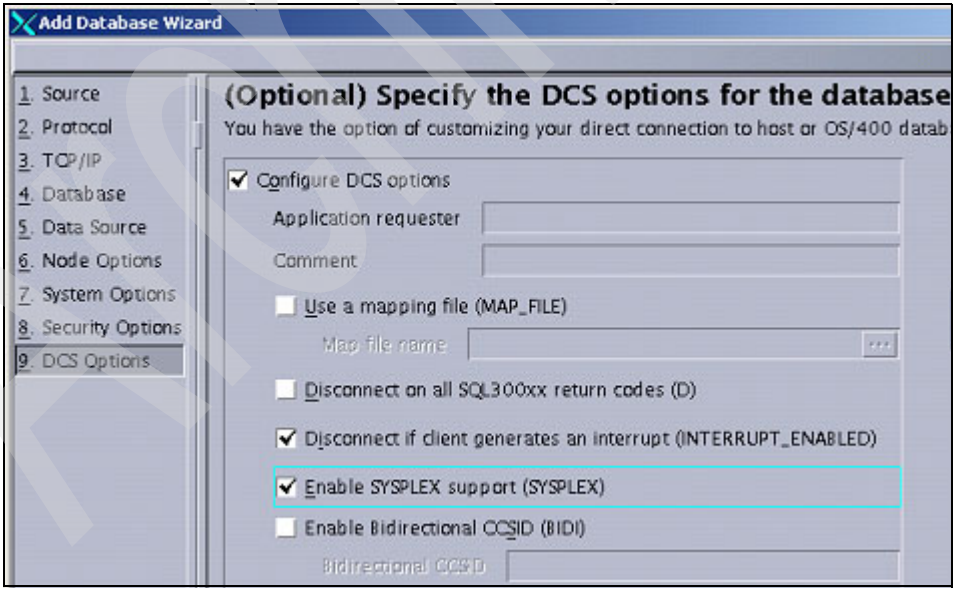

10. Enter `STLCEL4` for System name. Click **Next**.



11. Choose **Server authentication (SERVER)**.




12. On the Specify the DCS options for the database window, select **Disconnect if client generates an interrupt (INTERRUPT_ENABLED)** and **Enable SYSPLEX support (SYSPLEX)**. Then click **Finish**.

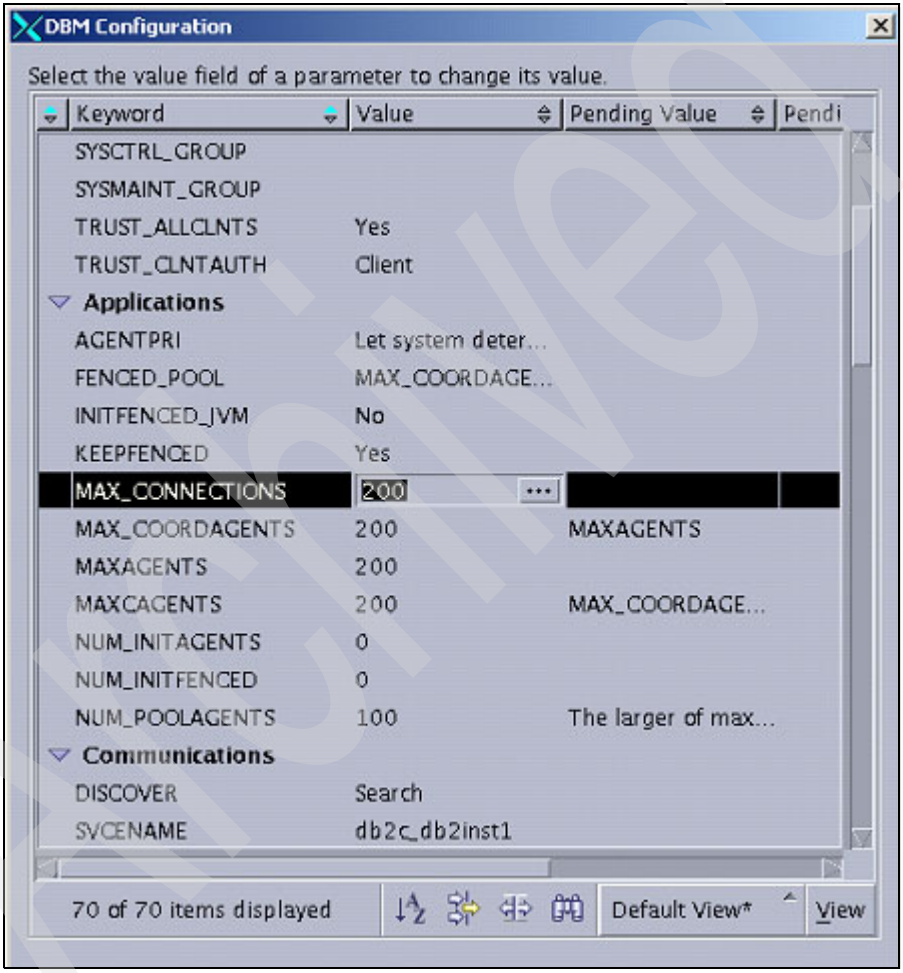



13. Test the connection by giving a user name and a password.

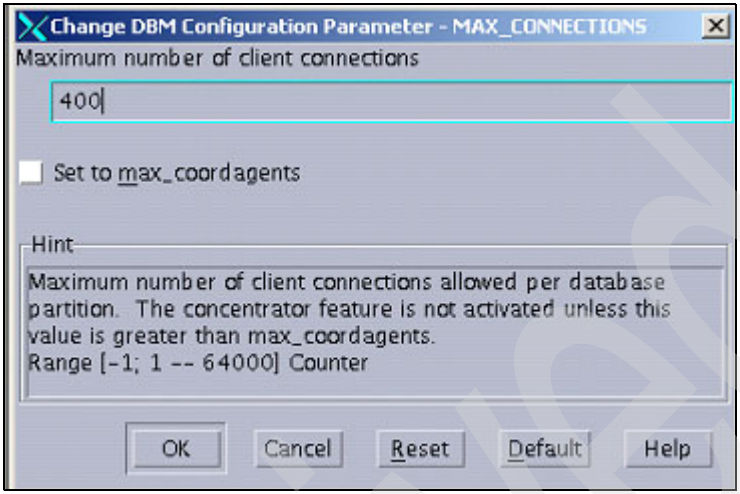
## Enable connection concentrator

Using DB2 Configuration Assistant (db2ca), do the following:

1. Launch **DB2 Client Configuration Assistant** either from the Start menu or by issuing the **db2ca** command from DB2 CLP.
2. From the menu, click **Configure** and choose **DBM Configuration**.
3. Highlight **MAX_CONNECTIONS**, to change value click the **...** button.



4. This opens the Change DBM Configuration Parameter - MAX_CONNECTIONS window.




5. Enter the new value and click **OK**. The new value of MAX_CONNECTIONS is shown as a Pending Value.

Restart DB2 Database Manager either from the DB2 Control Center or by issuing DB2 CLP commands as described in 4.2.2, “Using CLP” on page 78.

### 4.2.2 Using CLP

As an alternative to CCA, you may use the DB2 Command Line Processor (CLP) to configure DB2 Connect EE.

If you choose CLP, run the following commands:

- **db2set DB2CONNECT_IN_APP_PROCESS=NO**
- **Db2 catalog tcpip node SYSPLEX remote STLCEL4 server 8030 ostype os390**

  Note that SYSPLEX is an arbitrary name chosen for the node. The IP address can also be used instead of the host name.

  On z/OS, STLCEL4 is the host name of the database server and 8030 is the port number used for DB2. You can use the **-DISPLAY DDF** command to get the port number information.
- **Db2 catalog db db0g at node SYSPLEX authentication dcs**

  db0g is an arbitrary name. You use it to connect to DB2 for z/OS. You will use the same alias when you configure the PeopleSoft application server domain.
- **Db2 catalog dcs db db0g as DB0G parms ‘,,INTERRUPT_ENABLED,,,SYSPLEX,,,’**

  db0g is the name of the catalogued database, and DB0G is the location name of the DB2 subsystem on the host. Use **-DISPLAY DDF** for this information.The sixth positional parameter SYSPLEX makes DB2 Connect Sysplex aware.
- **Db2 terminate**
- **Db2 connect to db0g user<username> using <passwd>**

  The connection to the database is tested here. Before you issue this command, make sure that both DB2 members and both DDFs are started.
- **db2 update dbm cfg using MAX_CONNECTIONS <newvalue>**
- **$ db2 update dbm cfg using MAX_CONNECTIONS 400**
- To restart DB2 database manager:
  - **$ db2 db2stop**
  - **$ db2 db2start**
- To verify the changed value of MAX_CONNECTIONS, issue the following command and grep for MAX_CONNECTIONS:
  - **$ db2 get dbm cfg**

## 4.3 Application server domain configuration

When configuring the application server domain on the PeopleSoft application server machine, `DBNAME` should be specified as a data sharing group name. (Figure 4-2).

```
-----------------------------------------------
Quick-configure menu -- domain: DB0G
-----------------------------------------------
     Features                          Settings
    ==========                        ==========
  1) Pub/Sub Servers    : No    12) DBNAME      :[DB0G]
  2) Quick Servers      : No    13) DBTYPE      :[DB2ODBC]
  3) Query Servers      : No    14) UserId      :[QEDMO]
  4) Jolt               : Yes   15) UserPswd    :[QEDMO]
  5) Jolt Relay         : No    16) DomainID    :[PT81]
  6) PC Debugger        : No    17) AddToPATH   :[.]
  7) Opt Engines        : No    18) ConnectID   :[people]
  8) Event Notification: No     19) ConnectPswd:[people]
  9) MCF Servers        : No    20) ServerName  :[]
                                21) WSL Port    :[7000]
     Actions                    22) JSL Port    :[9000]
    =========                   23) JRAD Port   :[9100]
 10) Load config as shown
 11) Custom configuration
  h) Help for this menu
  q) Return to previous menu

HINT: Enter 12 to edit DBNAME, then 10 to load

Enter selection (1-23, h, or q): ▮
```

*Figure 4-2 Application server domain configuration: DBNAME:DB0G*

## 4.4 Validating Sysplex awareness setup in DB2 Connect

You can use these tools to validate the Sysplex awareness setup in DB2 Connect:

- DRDA trace
- Test program

### 4.4.1 DRDA trace

The DRDA trace shows which Sysplex members are known by WLM and participating in the load balancing.

DB2 Connect receives a prioritized list of Sysplex members from the WLM. Each Sysplex returns weighted priority information for each connection address. This list is then used by DB2 Connect to handle the incoming CONNECT requests by distributing them among the Sysplex members with the highest assigned

priorities. You can capture this data using DRDA trace and use it to verify your DB2 Connect and Sysplex data sharing setup.

The following process shows how to interpret DRDA trace data for a two-member data sharing group with IP addresses 192.138.103.170 and 192.138.103.175, using port 8030:

- Start the DRDA trace by using the **db2drdat on** command from DB2 Connect command line processor:

  ```
  $ db2drdat on -i -c -r -s
  ```

- Connect to Sysplex data sharing group DB0G:

  ```
  $ db2 connect to DB0G user usr1 using passwd usr1pswd
  ```

- Turn off the DRDA trace using the **db2drdat off** command:

  ```
  $ db2drdat off
  ```

Trace will be written to a file called db2drdat.dmp under the db2dump directory

Example 4-1 shows the snippet of DRDA trace when DB2 Connect is connected to a Sysplex. In the trace file look for the ACCDBRM receive buffer.

*Example 4-1 DRDA trace*

```
        RECEIVE BUFFER(AR):  ACCRDBRM RPYDSS       (ASCII)           (EBCDIC)
         0 1 2 3 4 5 6 7  8 9 A B C D E F   0123456789ABCDEF  0123456789ABCDEF
 0000  0073D0020001006D 2201000611490004   .s.....m"....I..  ..}...._........
 0010  000D002FD8E3C4E2 D8D3F3F7F0000C11   .../............  ....QTDSQL370...
 0020  2EC4E2D5F0F7F0F1 F1000A0035000611   ............5...  .DSN07011.......
 0030  9C0025000C11A0E6 C1E8D2C1D9404000   ..%..........@@.  .......WAYKAR  .
 0040  0621252434002E24 4E0006244C000200   .!%$4..$N..$L...  ........+...<...
 0050  24244D0006244F00 20000A11E8C0A867   $$M..$0. ......g  ..(...!.....Y{y.
 0060  AA1F5E0006244F00 1F000A11E8C0A867   ..^..$0........g  ..;...!.....Y{y.
 0070  AF1F5E                              ..^               ..;
```

You can parse the ACCDBRM reply as follows:

Starting at line 40, column 5 in receive buffer, you will see the following (all values are in Hexa):

```
Length = '002E'
Codepoint = '244E' (Server list)
   Length = '0006'
   Codepoint = '244C' (Server list count)
      Data = '0002' (Two entries)
   Length = '0024'
   Codepoint = '244D' (server list servers)
      Length.1 = '0006'
      Codepoint.1 = '244F' (Server priority weight, DB2 gets this from WLM)
```

```
Data.1 = '0020' (32)
Length.1 = '000A'
Codepoint.1 = '11E8' (IP address, made up of 4 byte address and 2
byte                  socket number)
Data.1 = 'C0A867AA1F5E' (192.138.103.170, port number 8030)
Length.2 = '0006'
Codepoint.2 = '244F' (Server priority weight, DB2 gets this from WLM)
Data.1 = '001F' (31, essentially equal weight but .170 address being
first in              this list will be chosen on next connection)
Length.1 = '000A'
Codepoint.1 = '11E8' (IP address, made up of 4 byte address and 2
byte                  socket number)
Data.1 = 'C0A867AF1F5E' (192.138.103.175, port number 8030)
```

You can use Server list count and IP addresses values to verify whether all members are participating in the data sharing environment.

### 4.4.2 Test program to check load balancing

You can also test whether load balancing works by running the test program we describe in Appendix B, “Test program to validate load balancing” on page 117.

5

# Functional test of load balancing

This chapter describes the preliminary functional tests run at the Silicon Valley Laboratory (SVL). The objective of this exercise is to test DB2 Connect connection concentrator and, more specifically, its load balancing function in a Sysplex data sharing environment with simple read-only PeopleSoft applications.

We discuss the following:

- Application workload used for the test
- Monitoring load balancing
- Monitoring failover

## 5.1 Application workload used for the test

The purpose was to test the functionality of the connection concentrator load balancing function. We used very simple read-only queries for this test. We used a subset of PeopleSoft Financials online applications including the following:

- Asset management
  - cost history listing
- Payables
  - Payment inquiry
  - Voucher inquiry
- Receivables
  - Inquire customer item

To simulate user workload we used the Microsoft® Web Application Stress Tool V1.1. We ramped up to 100 users for this preliminary functional test.

## 5.2 Monitoring load balancing

When connection concentrator is turned on, DB2 Connect can perform load balancing at the transaction boundary (commit/rollback). To observe this load balancing we monitored DB2 Connect using the DB2 Performance Expert (DB2PE) client on our workstation. With this monitoring tool, we could observe how the application thread execution switches between member DB1G and member DB2G when the CPU on either member gets loaded.

Figure 5-1 on page 85 gives DB2PE snapshots showing connection assignment & SQL activity on two members when both members are equally busy. The connection used by the PSAPPSRV process is executing on member DB2G, and the connection used by PSSAMSRV is executing on member DB1G.

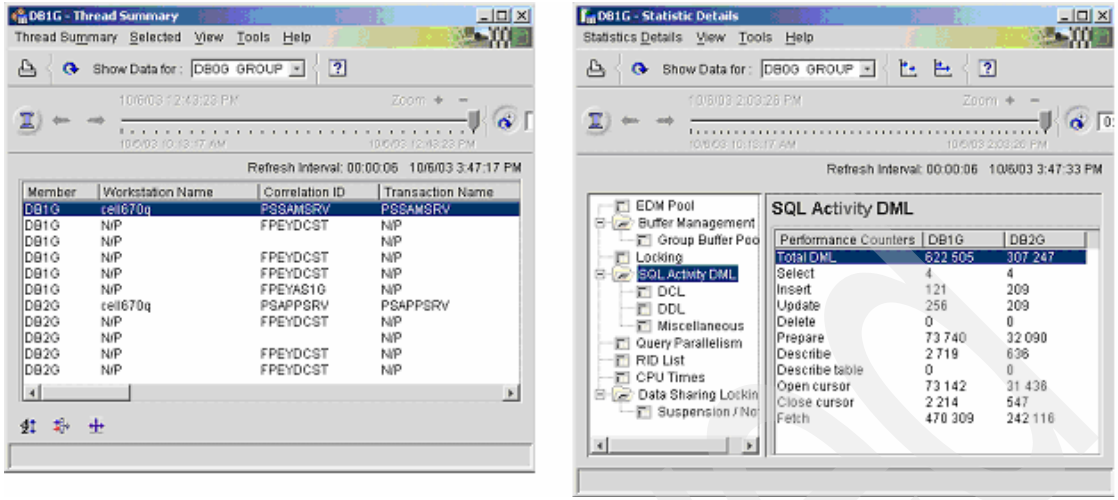



*Figure 5-1 50% CPU activity on each member DB1G and DB2G*

Now, if we load member DB2G with CPU-intensive jobs (100% in this case), we observe that the database activity moves from member DB2G to member DB1G.

DB2 Connect creates new connections on member DB1G and moves the workload from DB2G to DB1G, as shown in Figure 5-2.

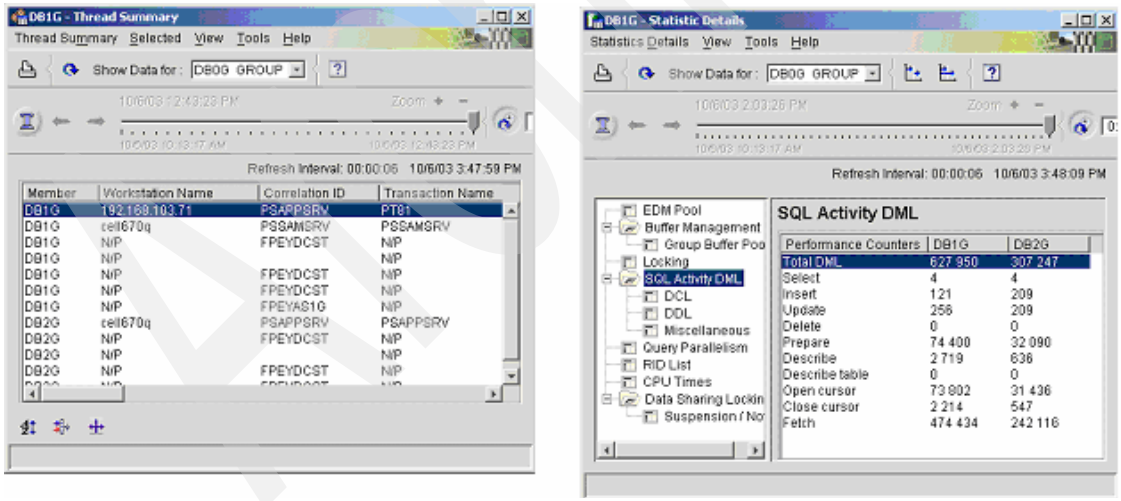



*Figure 5-2 100% CPU activity on both member DB1G and DB2G*

## 5.3 Monitoring failover

Connection concentrator also helps to move connections between members in case of one member's hardware failure. Figure 5-3 shows DB2PE snapshots of connection assignment between two members after the application server starts.

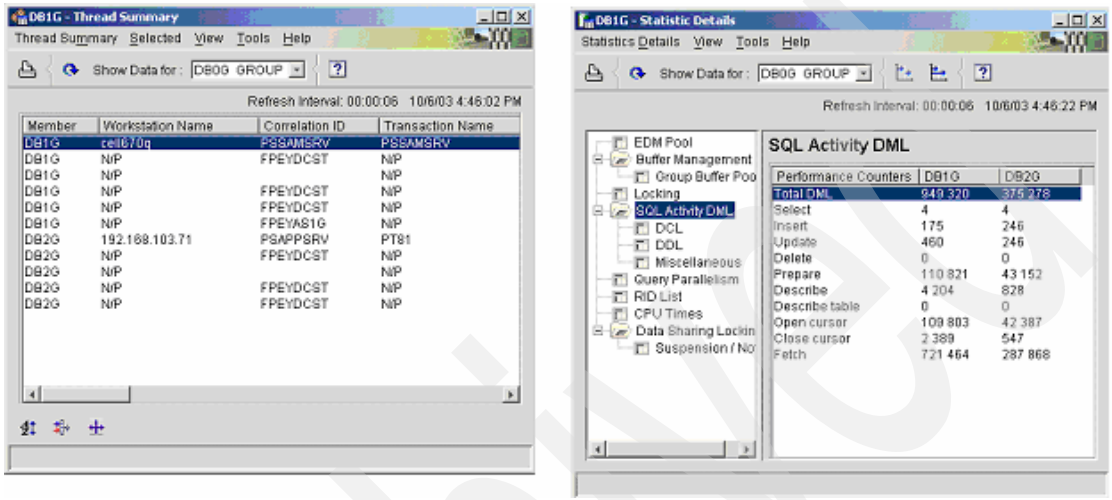
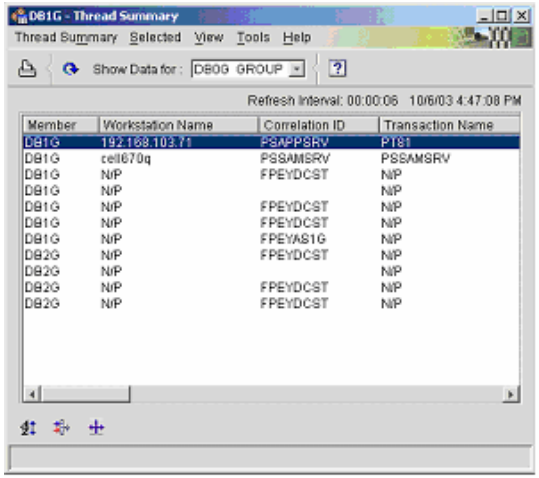



*Figure 5-3 Connection assignments when the application server starts up*

If we abruptly bring down DB2G, the connection used by PSAPPSRV moves to DB1G at the next application server request (Figure 5-4 on page 87 and Figure 5-5 on page 87).



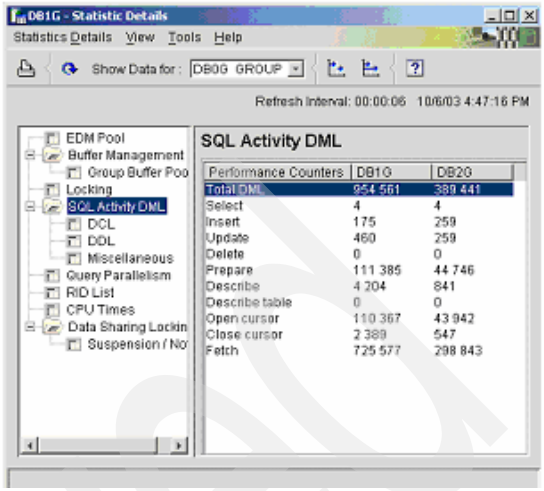
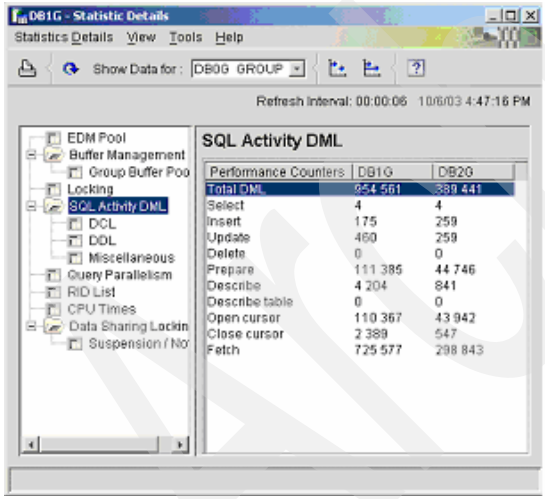




*Figure 5-4 Failover from DB2G to DB1G*

SQL activity is only on DB1G.



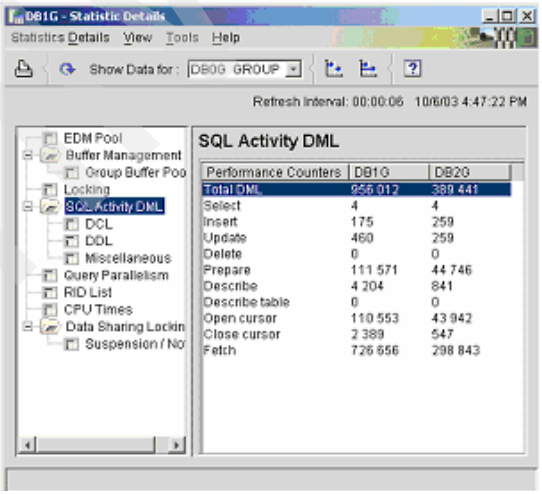
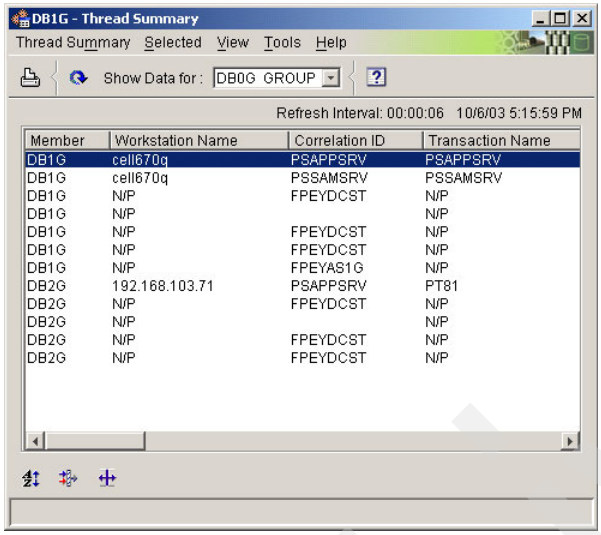




*Figure 5-5 SQL activity on DB1G*

Now if we bring up member DB1G, we observe that the connection used by PSAPPSRV moves back to DB2G at the next application server request (Figure 5-6).




| Member | Workstation Name | Correlation ID | Transaction Name |
|---|---|---|---|
| DB1G | cell670q | PSAPPSRV | PSAPPSRV |
| DB1G | cell670q | PSSAMSRV | PSSAMSRV |
| DB1G | N/P | FPEYDCST | N/P |
| DB1G | N/P | | N/P |
| DB1G | N/P | FPEYDCST | N/P |
| DB1G | N/P | FPEYDCST | N/P |
| DB1G | N/P | FPEYAS1G | N/P |
| DB2G | 192.168.103.71 | PSAPPSRV | PT81 |
| DB2G | N/P | FPEYDCST | N/P |
| DB2G | N/P | | N/P |
| DB2G | N/P | FPEYDCST | N/P |
| DB2G | N/P | FPEYDCST | N/P |

*Figure 5-6 Connection moves back to DB2G*

6

# Volume tests of load balancing

A second series of more complex tests for load balancing were done at Silicon Valley Labs using a higher number of connected users (400) and more complex transactions including updates and inserts.

We run two scenarios of load balancing tests:

- Two members running at the same time; one member gets CPU-saturated.
- One member runs first with all of the workload, than the second member is activated.

## 6.1 Load balancing: CPU saturation

In this test we have 400 (simulated) users and 15 application servers. Members DB1G and DB2G are active at the same time and process workload. We use the same subset of PeopleSoft Financials online applications as we did in Chapter 5, “Functional test of load balancing” on page 83 but this time with a more complex workload including updates and inserts.

After 15 minutes of steady state, member DB2G gets very busy with 100% CPU usage. WLM is aware of that, and when it sends the connection list to DB2 Connect, it points the connection concentrator to the least busy member (DB1G) where more processing resources are available. DB2 Connect connection concentrator directs the incoming workload to DB1G.

Figure 6-1 shows the time frame where DB2G saturates the CPU at 100%.

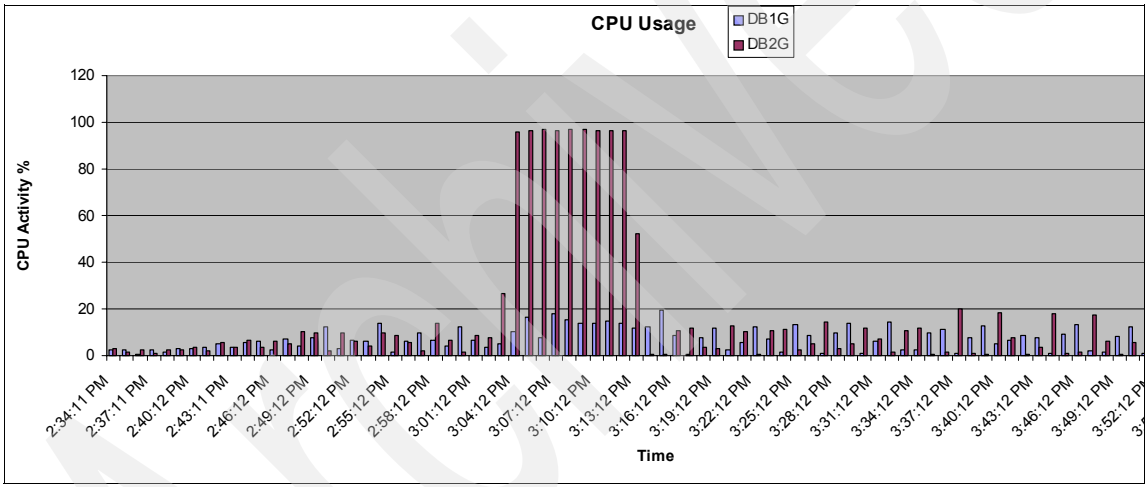



*Figure 6-1 CPU usage by members DB1G and DB2G*

Figure 6-2 shows that during the same time frame there is no more PeopleSoft SQL activity happening on DB2G. This is because DB2 Connect stopped sending PeopleSoft work to this member as it has insufficient available CPU resources. Instead, DB1G executes all SQL activity during that time frame. DB2G will pick up SQL activity again when it has more processing resources available.

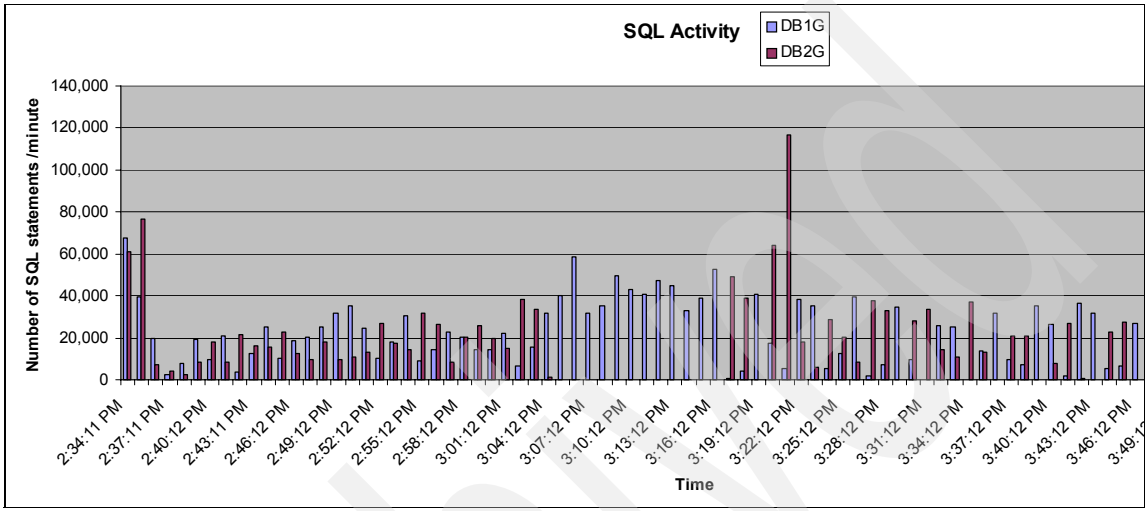



*Figure 6-2 PeopleSoft SQL activity on members DB1G and DB2G*

## 6.2 Load balancing: additional member

This test starts with only one member, DB1G. After about 10 minutes of steady state a new member, DB2G, is introduced in the data sharing environment. DB2 Connect reacts to this change in environment by spawning new connections on member DB2G and moving some of the load from DB1G to DB2G.

Figure 6-3 shows how the SQL workload splits when a new available member is introduced in the data sharing group.

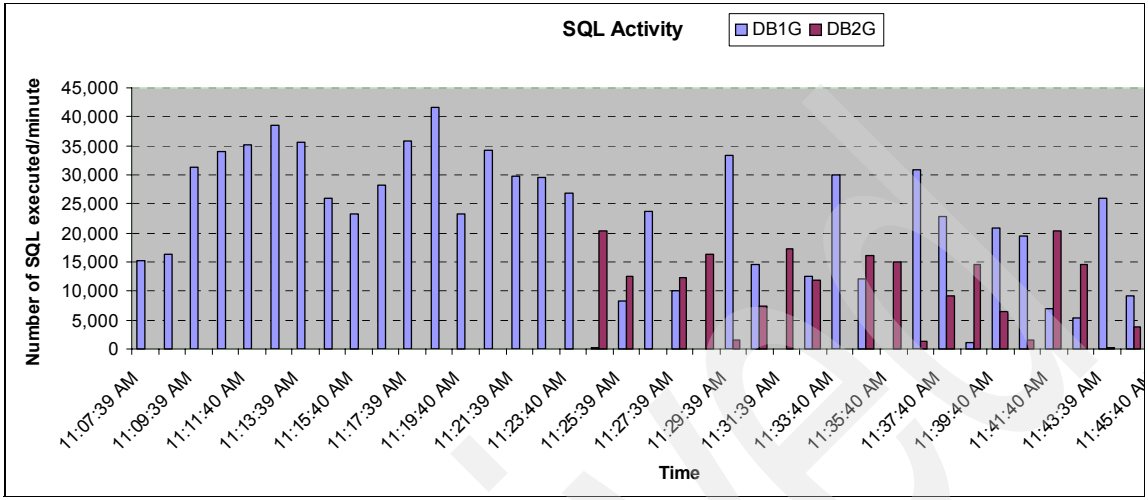



*Figure 6-3 PeopleSoft SQL activity on single member first, both members next*

Figure 6-4 shows the number of connected threads. First, DB1G has 23 connected threads to handle all of the workload. Next, when the second member DB2G is introduced, 13 threads are created on this member and the activity splits between the two members. DB1G has 13 active threads. DB1G still has 23 threads, but half are inactive and half are active.

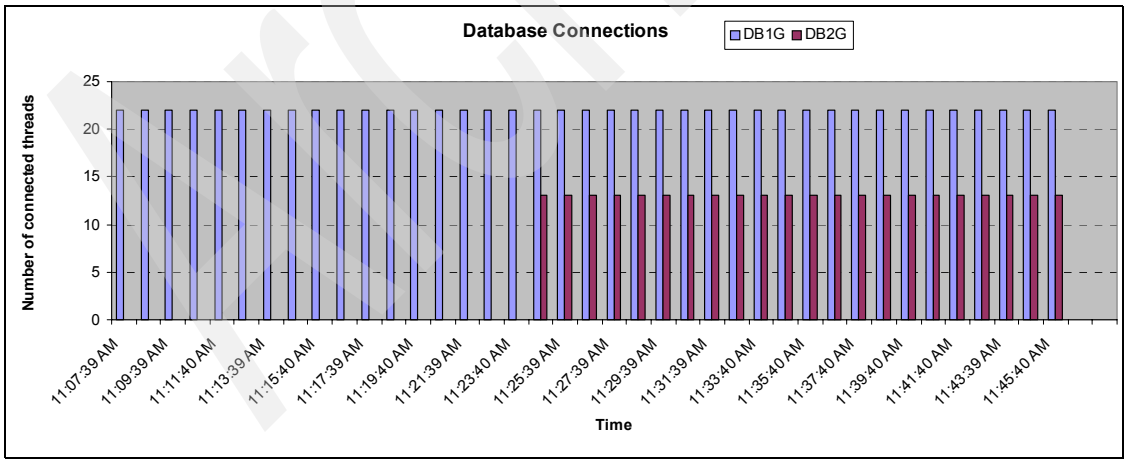



*Figure 6-4 Database connections on members DB1G and DB2G*

Figure 6-5 shows the CPU activity. The CPU is busy on DB1G when this member is the only one to execute the whole workload. CPU usage splits when two members share the workload.

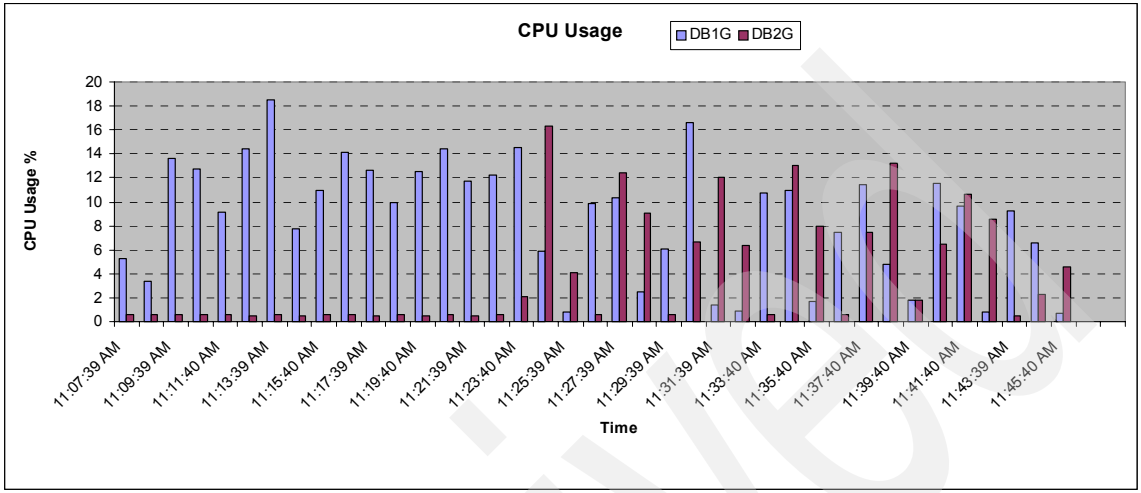



*Figure 6-5 CPU usage on members DB1G and DB2G*

All of these tests show the effective management of load balancing by DB2 Connect connection concentrator.

7

# ESS performance considerations for PeopleSoft

This chapter provides recommendations regarding ESS and ways applications may benefit from the enablement of ESS features and functions.

We discuss the following topics:

- Introducing ESS
- Parallel Access Volumes (PAV)
- FlashCopy®
- I/O Priority Queuing
- FICON™

## 7.1 Introducing ESS

The IBM TotalStorage® Enterprise Storage Server® (ESS) is the most powerful IBM disk storage server, developed using IBM Seascape® architecture. The ESS provides unmatchable functions for the entire family of e-business servers, and also for non-IBM (that is, Intel®-based and UNIX®-based) families of servers. Across all of these environments, the ESS features unique capabilities that enable it to meet the most demanding requirements of performance, capacity, and data availability that the computing business may require.

The Seascape architecture is the key to the development of IBM storage products. Seascape enables IBM to take the best of the technologies developed by the many IBM laboratories and integrate them, producing flexible and upgradeable storage solutions. This Seascape architecture design has enabled the IBM TotalStorage Enterprise Storage Server to evolve from the initial E models to the succeeding F models, and to the recently announced 800 models, each featuring new, more powerful hardware and functional enhancements and always integrated under the same successful architecture with which the ESS was originally conceived.

The move to e-business presents companies with both extraordinary opportunities and significant challenges. A whole new world of potential customers, automated and streamlined processes, and new revenue streams are being fueled by e-business. Consequently, companies also face an increase of critical requirements for more information that is universally available online, around the clock, every day of the year.

To meet the unique requirements of e-business, where massive swings in the demands placed on your systems are common and continuous operation is imperative, you need very high-performance, intelligent storage technologies and systems that can support any server application in your business, today and into the future. The IBM TotalStorage Enterprise Storage Server has set new standards in function, performance, and scalability in these most challenging environments.

### 7.1.1 The IBM TotalStorage Enterprise Storage Server Model 800

Since its initial availability with the ESS Models E10 and E20, and then with the succeeding F10 and F20 models, the ESS has been the storage server solution offering exceptional performance, extraordinary capacity, scalability, heterogeneous server connectivity, and an extensive suite of advanced functions to support customers' mission-critical, high-availability, multi-platform environments. The ESS set a new standard for storage servers when it was first available in 1999, and since then it has evolved into the F models and the

recently announced third-generation ESS Model 800, keeping up with the pace of customers' needs by adding more sophisticated functions to the initial set, enhancing connectivity options, and powering its performance features.

The IBM TotalStorage Enterprise Storage Server Model 800 provides significantly improved levels of performance, throughput, and scalability while continuing to exploit the innovative features introduced with its preceding E and F models such as Parallel Access Volumes, Multiple Allegiance, I/O Priority Queuing, the remote copy functions (synchronous and asynchronous), and the FlashCopy point-in-time copy function. Also the heterogeneous server support characteristic—for connectivity and remote copy functions—of previous models is continued with the ESS Model 800, in addition to the enhancement features that were introduced more recently with the F models, such as disk capacity intermix, ESS Master Console, S/390 CUIR, 32 K cylinder large-volume support, 72.8 GB disk drives, new 18.2 GB and 36.4 GB 15000 RPM disk drives, non-synchronous PPRC Extended Distance, and the additional connectivity options for the remote copy functions.

The new IBM TotalStorage Enterprise Storage Server Model 800 introduces important changes that dramatically improve the overall value of ESS in the marketplace and provide a strong base for strategic Storage Area Network (SAN) initiatives.

Among the enhancements introduced in the Model 800 are:

- 2 GB Fibre Channel/FICON host adapter
- Up to 64 GB of cache
- New, more powerful SMP cluster processors with a Turbo feature option
- 2 GB non-volatile storage (NVS) with double the bandwidth
- A doubling of the bandwidth of the Common Platform Interconnect (CPI)
  RAID 10 array configuration capability

These hardware enhancements introduced by the new IBM TotalStorage Enterprise Storage Server Model 800 all combine to provide a balanced two-fold performance boost as compared to the predecessor F models, and up to two-and-a-half boost with the Turbo processor option.

PeopleSoft has been using ESS as a basic part of its zSeries benchmarking support. In June of 2003, they installed a Model 800 in Pleasanton. It replaced a Model F20, which had been in use for a year prior to this. The Model 800 has the following features:

- 12 × 2144 Disk Eight-Pack - 72.8 GB
- 1 × 2717 ESS Master Console
- 4 × 3013 64-BIT ESCON Host Adapter
- 8 × 3025 2 GB Fibre Channel/FICON Short Wave
- 1 × 3604 Standard Processor

- 1 × 4016 32 GB Cache
- 1 × 8004 PAV - up to 5 TB
- 1 × 8306 FlashCopy - up to 8 TB

Since it was installed, it has been used in all zSeries benchmarks conducted by PeopleSoft.

The following sections discuss ESS features such as Parallel Access Volume (PAV), FlashCopy, I/O priority queuing, and FICON. All of these features increase PeopleSoft performance on zSeries by significantly improving data access time.

## 7.2 Parallel Access Volume

PAV is a performance feature. It enables multiple concurrent non-conflicting I/Os to a single DASD volume. As a result, IOSQ time (the time the UCB in OS/390 is blocked for other I/Os directed to the same volume because the UCB is busy) is highly reduced.

Parallel Access Volume (PAV) is one of the original features that the IBM TotalStorage Enterprise Storage Server provides specifically for z/OS users. Simply stated, PAV enables multiple concurrent I/Os to the same volume at the same time from applications running on the same z/OS system image. This concurrency helps zSeries applications better share the same logical volumes with reduced contention. The ability to send multiple concurrent I/O requests to the same volume nearly eliminates I/O queuing in the operating system, thus reducing I/O response times.

Traditionally, access to highly active volumes has involved manual tuning, splitting data across multiple volumes, and more in order to avoid those hot spots. With PAV and the z/OS Workload Manager, you can now almost forget about manual performance tuning. The Workload Manager can tune your PAV configuration automatically and adjust it to workload changes. The ESS in conjunction with z/OS has the ability to meet the highest performance requirements.

### 7.2.1 PAV Characteristics

The IBM TotalStorage Enterprise Storage Server Model 800 is a modern disk storage subsystem with large cache sizes and disk drives arranged in RAID arrays (ranks). I/O solved in cache is much faster than I/O solved on disk, as no mechanical parts (actuator) are involved. Plus, I/Os can take place in parallel, even to the same volume. This is true for reads, and it is also possible for writes, as long as different extents on the volume are accessed. The ESS emulates zSeries CKD volumes over RAID-5 and RAID-10 disk arrays. While the z/OS

operating system continues to work with these logical DASD volumes as a unit, its tracks are spread over several physical disk drives. So parallel I/O from different applications to the same logical volume would also be possible for cache misses (when the I/Os have to go to the disk drives, involving mechanical movement of actuators) as long as the logical tracks are on different physical disk drives.

Traditionally the operating system does not attempt to start more than one I/O operation at a time to a logical device. If an application initiates an I/O to a device that is already busy, the I/O request will be queued by the I/O Supervisor (IOS) component of z/OS until the device completes the other I/O and becomes available. z/OS systems queue I/O activity on a Unit Control Block (UCB) that represents the logical device. When volumes are treated as a single resource and serially reused, high I/O activity could adversely affect performance. This could result in large IOS queuing (IOSQ) times, traditionally a major component of z/OS response time. This queue time component can be eliminated if the devices are defined with PAVs.

To take advantage of the ESS capability to process multiple concurrent I/Os to a logical volume, the ESS and z/OS together support the PAV feature that enables parallel I/Os to a volume from one z/OS system image. PAV is an optional feature of the ESS.

The PAV implementation introduces the new concept of *alias device*, in addition to the conventional *base device*. The alias devices provide the mechanism for z/OS to initiate parallel I/Os to a volume.

Each device, base and alias alike, has a unique address in the ESS control unit image. The unit addresses range from x'00' to x'FF'. Each device has its own subchannel, UCB, and device address on the host. The base device's address is the actual unit address of the volume. There is one base address per volume.

An alias device does not have any physical space of its own associated with it. It simply maps to a base and enables a z/OS host to use several UCBs for the same logical volume instead of one UCB per logical volume. For example, base address 2C00 may have alias addresses 2CFD, 2CFE, and 2CFF associated with it. Each UCB supports one parallel I/O operation as before, but the three aliases in addition to the base now allow for four parallel I/O operations to the same volume.

The definition and exploitation of PAVs requires the operating system software to define and manage alias device addresses and subchannels. The z/OS operating system has this support and can issue multiple channel programs to a logical volume, enabling simultaneous access to the volume by multiple users or jobs. Reads can be satisfied simultaneously, as well as writes to different domains. The domain of an I/O consists of the specified extents to which the I/O

operation applies. Writes to the same domain still have to be serialized to maintain data integrity. Both base and alias devices are defined on the ESS using the ESS Specialist. They must be accompanied by matching hardware configuration definition (HCD) device definitions on the zSeries server. In the HCD, you specify which device addresses are bases, and which are aliases.

**Important:** The device definitions in the ESS and HCD must match.

A base device can be associated with aliases only from the same ESS logical subsystem (LSS). In other words, an alias can be assigned only to a base in the same LSS. The maximum number of devices per LSS is 256, including both bases and aliases. Theoretically, you could define one base and 255 aliases in an LSS. At the other extreme, you could define 256 base devices and no aliases. Typically, you will have a mix of both. The number of base devices you define depends on the amount of installed storage capacity in the LSS, and the size of the volumes you define. To provide enough concurrency and performance for the base devices, you then must define and assign an adequate number of aliases to the bases.

Together the base and aliases are called exposures. We also use the terms *parallel addresses*, or PAV addresses (simply PAVs), when referring to a base and its aliases together. Thus the number of PAVs for a volume is one plus the number of aliases assigned to it.

### 7.2.2 Dynamic and static PAVs

If you have decided to take advantage of the performance benefits to be obtained from using PAVs, then you should consider the type of PAV support to implement and the number of PAVs to define.

When initially defined, aliases are assigned to a certain base device. An alias can later be reassigned to another base while the devices remain online. Ideally, if you had a large number of aliases, you could give each base enough aliases at the beginning so that the devices would never experience I/O Supervisor (IOS) queuing delays. More typically, you will not have that many aliases to work with; therefore, the alias assignments must be modified to adapt to changes in workload that occur during daily workload shifts and in the long run. Because the number of devices per LSS is limited to 256, aliases can be a scarce resource that must be managed to maximize performance. For best I/O management, the aliases should be assigned to the base devices based on their needs. The busiest base devices at any given moment, or those handling the most important workload, should have more aliases than the inactive bases.

It will not always be easy to predict which volumes should have an alias address assigned and how many. The workload intensity of volumes can change over time, sometimes so rapidly that manual tuning cannot keep up with the changes in volume activity. Particularly in cases when the number of aliases is limited, it is important that aliases can be assigned to the volumes needing them most.

You can reassign aliases manually using the ESS Specialist. However, this is a labor-intensive process and, to achieve optimum results, should be repeated whenever there are workload changes. The process of alias reassignment must be automatic and sensitive to the importance and level of I/O activity on the device. Instead of manually tuning your aliases, you can let z/OS Workload Manager (WLM) manage the aliases automatically according to your goals. The WLM cooperates with the IOS to allow for the automatic management of aliases. This function is called *dynamic alias management*. With dynamic alias management, WLM can automatically perform alias device reassignments from one base device to another to help work meet its goals and to minimize IOS queueing as workloads change.

WLM manages PAVs across all members of a sysplex. When making decisions on alias reassignment, WLM considers I/O from all systems in the sysplex. By default, the function is turned off, and must be activated explicitly for the sysplex through an option in the WLM service definition, and through a device level option in HCD. Dynamic alias management requires your sysplex to run in WLM Goal mode.

Devices that are enabled for dynamic alias management are called *dynamic* PAVs. The alternative is *static* PAVs. With static PAVs, the association of aliases to a base device is predefined through the ESS Specialist. Static alias assignment can only be changed using the ESS Specialist. It is possible to define both dynamic and static PAVs on an LSS, if necessary.

## Enabling dynamic PAV

You can enable or disable dynamic alias management on a device-by-device basis with the WLMPAV parameter in HCD. If WLMPAV=YES, then the aliases of the device will be managed dynamically by WLM. If WLMPAV=NO, then the aliases of the device are static. The default for each device is YES.

Besides being enabled for each volume through the HCD, dynamic alias management must also be enabled globally in the sysplex. This is done by setting the Dynamic alias management service definition option to YES on the WLM Service Coefficient/Service Definition Options panel. The default is NO (dynamic alias management is disabled).

The I/O priority management option on the WLM Service Coefficient/Service Definition Options panel activates ESS I/O Priority Queuing (see 7.4, “I/O priority

queuing” on page 108). At the same time, it also determines which of two dynamic alias management mechanisms will be used, when dynamic alias management is enabled. If the option is set to YES, the goal-based mechanism for dynamic alias management is used. If set to NO (the default), the efficiency-based mechanism is used. See “WLM dynamic alias management” on page 102.

Dynamic aliases effectively form a pool from which WLM can choose aliases to assign to the base devices needing them most. This makes your configuration planning easier, as you do not have to plan in advance how many aliases each and every volume should have. Rather, you only need to plan how many aliases in total there will be available in each LSS.

We recommend always using dynamic PAVs if possible. You should consider using static PAVs only in exception cases, such as:

- Your system is not running in WLM Goal mode. (Note that z/OS V1R3 and later only run in Goal mode.)
- You are sharing devices between sysplexes.

### WLM dynamic alias management

When dynamic alias management is enabled, the Workload Manager monitors the device performance and is able to dynamically reassign alias devices from one base to another if predefined goals for a workload are not met. The WLM instructs the IOS to reassign an alias.

WLM also keeps track of the devices utilized by the different workloads, accumulates this information over time, and broadcasts it to the other systems in the same sysplex. If WLM determines that any workload is not meeting its goal due to IOSQ time, WLM will attempt to find an alias device that can be reallocated to help this workload achieve its goal.

The use of dynamic PAVs requires that devices assigned as eligible for dynamic PAV management must not be shared by systems outside of the sysplex. WLM does not consider I/O from systems outside a sysplex when reassigning aliases.

There are two different mechanisms that WLM uses to tune the alias assignment:

- The first mechanism is goal-based. This logic attempts to give additional aliases to a PAV-enabled device that is experiencing IOS queue delays and is affecting a service class period that is missing its goal. To give additional aliases to the receiver device, a donor device must be found with a less important service class period. A bitmap is maintained with each PAV device that indicates the service classes using the device. The objective of a goal algorithm is to help a service class period meet its goal.

- The second mechanism is efficiency-based. This algorithm moves aliases to high-contention PAV-enabled devices from low-contention PAV devices. This tuning is based on efficiency rather than directly helping a workload to meet its goal. The objective of an efficiency algorithm is to reduce overall queuing in the system.

Because adjusting the number of aliases for a PAV-enabled device affects any system using the device, a sysplex-wide view of performance data is important and is needed by the adjustment algorithms. Each system in the sysplex broadcasts local performance data to the rest of the sysplex. By combining the data received from other systems with local data, WLM can build the sysplex view.

The efficiency algorithm uses IOS queue length (the average number of I/O requests in the IOS queue) of base devices to determine when aliases should be moved. Devices whose average IOS queue length exceeds 0.5 will be assigned more aliases to help the receiver devices. Devices whose average queue length is below 0.5 but above 0.05 will be assigned more aliases only if there are aliases available on idle or low-activity devices. (See APAR OW48647 for more information.)

If the goal mechanism is used, WLM will first take workload goals into consideration when moving aliases, then use volume queue lengths to determine which volumes within a service class should be assigned more aliases.

The efficiency algorithm scans the base devices once per minute to find devices that exceed thresholds and then reassigns aliases if necessary. The goal algorithm scans devices every 10 seconds, but a device that received or donated an alias is not touched for the next minute. WLM only moves one alias at a time to or from a base device. With the goal algorithm, multiple base devices of a service class can receive an alias during one interval.

Aliases of an offline device are considered *unbound*. WLM uses unbound aliases as the best donor devices. If you run with a device offline to some systems and online to others, you should make the device ineligible for WLM dynamic alias management in HCD.

RMF reports the number of PAVs for each device in its Monitor/DASD Activity report and in its Monitor II and Monitor III Device reports. RMF also reports which devices had a change in the number of PAVs during the RMF interval.

## PAV performance considerations

Our first recommendation is to always configure one alias for every base device as a minimum. There is a point when the benefits of adding more aliases starts to be less important.

There is limited value in configuring more PAVs than you have ESCON® channel paths to a system, as the number of ESCON channel paths becomes the limiting factor in data transfer.

Refer to *IBM TotalStorage Enterprise Storage Server Model 800*, SG24-6424, for more information about tuning PAVs.

## 7.3 FlashCopy

Today, data processing centers increasingly require applications to be available 24 hours per day, seven days per week, without compromising recoverability in the event of a failure.

FlashCopy enables the user to implement the following solutions:

- Efficient and seamless volume backup
- System or application cloning
- System or application testing
- Restarting of database applications
  a. Copy back FlashCopy target
  b. Apply logs to bring the database current

FlashCopy provides the ability to create point-in-time (PiT) copies. As soon as the FlashCopy command is processed, both the source and target volumes are available for application use. Only a minimal interruption is required for the FlashCopy relationship to be established, so the copy operation can be initiated. The copy is then created under the covers by the IBM TotalStorage Enterprise Storage Server (ESS), with minimal impact on other ESS activities.

FlashCopy is an optional feature that must be enabled in the ESS.

FlashCopy may also be used in conjunction with either the local or remote copies of data created by Peer-to-Peer Remote Copy (PPRC) and Extended Remote Copy (XRC), making it easy to create additional copies for rapid recovery following application errors or for other uses.

### 7.3.1 Characteristics

FlashCopy provides the ability to create point-in-time copies. As soon as the FlashCopy command is processed, both the source and target volumes are available for application use.

Characteristics of FlashCopy V1 include:

- FlashCopy V1 works at volume level.
- The source and target volumes must be in the same ESS logical subsystem (LSS).
- A source and a target volume can be involved in only one FlashCopy relationship at a time.

With FlashCopy Version 2, the previous characteristics change and new functionality has also been added:

- FlashCopy V2 can be used for dataset copies as well as volume copies.
- The source and target FlashCopy can be on different LSSs within an ESS.
- Multiple FlashCopy relationships are allowed.
- Incremental copies are possible.
- Inband commands can be sent over PPRC links to a remote site.
- FlashCopy consistency groups can be created.

FlashCopy V2 supports all FlashCopy V1 functions plus the preceding list of enhancements.

In addition, there has been a reduction in the FlashCopy establish times: Performance improvements in FlashCopy V2 provide a reduction of up to 10x in the time required to complete a FlashCopy establish command. With this significant reduction in establish time, operational interruption is further minimized and the benefits of FlashCopy can be extended to new application environments.

### 7.3.2 Database cloning

Business has a critical need for quick, accurate DB2 backup to support both production copy and recovery and development copy and restoration requirements.The process for recovery and restoration of entire DB2 subsystems often required the better part of a day. Moreover, there had to be a target LPAR for the new subsystem distinct from the LPAR of the source subsystem.

Entire production DB2 subsystems can be copied in mere minutes to a new standalone DB2 subsystem. This new subsystem offloads MIS functions, traditional Image Copies, and even Update activity (synchronization not discussed herein), helping to free the production system to devote more time to business-critical jobs. This process is extremely quick and far more efficient than traditional Image Copy Restores.

In addition, both production and developmental DB2 subsystems can be copied to offline volumes as static data and used to quickly restore the original subsystem.

As an experiment to reduce downtime for PeopleSoft applications, an effort was made to explore using FlashCopy to clone a Payroll database with the use of Mainstar's MS/VCR product to enable the database to be brought back online within the originating z/OS system image.

A white paper describes the data cloning process using FlashCopy and Mainstar with a PeopleSoft Payroll database. It can be found at:

http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100334

## Mainstar MS/VCR

As previously described, FlashCopy is a feature of the ESS. It provides an instant or point-in-time (PiT) copy of a logical volume giving an instantaneous copy or view of what the original data looked like at a specific point-in-time – sometimes called a time zero (T0) copy. FlashCopy allows accessing both the source and target volumes even before the physical copy process has completed. By creating an instant image of the source, FlashCopy enables applications using either the source or target copy to operate with only the minimal interruption while the FlashCopy relationship is being established.

Mainstar Mirroring Solutions / Volume Conflict Rename (MS/VCR) software gives access to “cloned” datasets on FlashCopied volumes. These clones have an inherent limitation: Although the target volume label reflects either the source or one specified for the target, the internal data—VTOC, VTOCIX, VVDS, and dataset names—all reflect the source volume label. MS/VCR is designed to rename and catalog the cloned data and fix the VTOC, VTOCIX, and VVDS.

Additionally, MS/VCR updates the DB2 internal control information. This is done using the DB2UPDATE component that conditions the cloned DB2 subsystem by modifying the directory, BSDSs, and catalog.

## Copy process

You can quickly copy an entire DB2 subsystem as a new subsystem in nine steps. It is recommended that the reader have a working knowledge of JCL and DB2. The steps are described in Appendix C, “FlashCopy using Mainstar MS/VCR” on page 125, in the sequence they must be executed.

1. BCSCLEAN
2. Stop source DB2
3. FlashCopy
4. Restart source DB2
5. COPYCHECK
6. RENAME
7. DB2UPDATE
8. DSNJ003
9. Start target DB2

## Copy metrics

Figure 7-1 shows the run times of the three steps needed to FlashCopy 133 volumes of data and bring up the source DB2 subsystem. Notice that the FLASHCOPY step is, in essence, completed for 133 volumes in just 4 minutes and the source DB2 subsystem is down for just 5 minutes. This is remarkable, considering the amount of data copied.

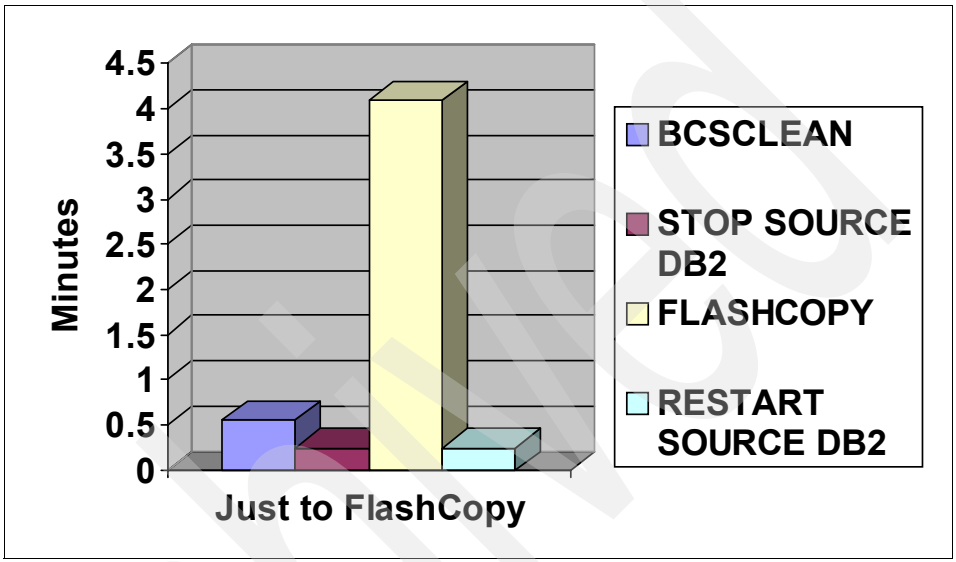



*Figure 7-1 FlashCopy only*

Figure 7-2 on page 108 shows the total run time to copy the data and to bring up a cloned DB2 subsystem in the same LPAR. This totals 36.6 minutes and shows all of the tasks needed to bring up an entirely new DB2 subsystem. Note that the COPYCHECK time is optional, but we choose to wait until the background copies completed in order to isolate the background copy RMF data. The RENAME job can be started as soon as the FlashCopy job is complete. This reduces the total run time to just 9.6 minutes.

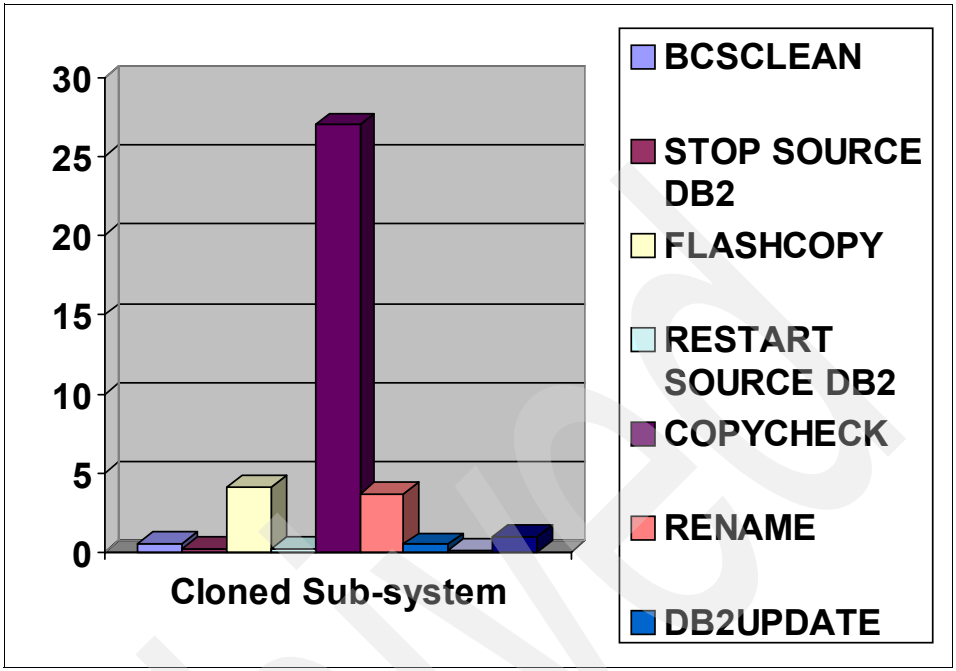



*Figure 7-2 Cloned DB2 subsystem*

## 7.4 I/O priority queuing

The IBM TotalStorage Enterprise Storage Server can manage multiple channel programs concurrently, as long as the data accessed by one channel program is not altered by another channel program.

If I/Os cannot run in parallel, for example, due to extent conflicts and must be serialized to ensure data consistency, the ESS will internally queue I/Os. Contrast this with the less-efficient procedure used by traditional disk storage subsystems that responded by posting a device-busy status to the operating system, which then had to re-drive channel programs.

This subsystem I/O queuing capability provides significant benefits:

- Compared to the traditional approach, I/O queuing in the storage subsystem eliminates the overheads associated with posting status indicators and re-driving channel programs.
- Channel programs that cannot execute in parallel are processed in the order they are queued. A fast system cannot monopolize access to a device also accessed from a slower system. Each system gets a fair share.

The ESS can queue I/Os from different z/OS system images in a priority order. z/OS Workload Manager can make use of this and prioritize I/Os from one system against the others. You can activate I/O Priority Queuing in WLM Goal mode with the I/O priority management option in the WLM Service Definition settings.

When a channel program with a higher priority comes in and is put in front of the queue of channel programs with lower priority, the priority of the low-priority programs is also increased. This prevents high-priority channel programs from dominating lower priority ones and gives each system a fair share.

## 7.5 FICON

Since the installation of the Model 800, PeopleSoft has seen a major improvement in disk performance. Part of this is due to the ESS improvements when compared to the Model F20. However, transfer time and overlapping of I/O requests due to the use of FICON compared to ESCON has made a significant difference.

FICON extends the ability of the IBM TotalStorage Enterprise Storage Server Model 800 to deliver bandwidth potential to the volumes needing it, when they need it.

For the z/Architecture™ and s/390 servers, the FICON attachment provides many improvements:

- Increased number of concurrent I/O connections over the link: FICON provides channel-to-ESS multiple concurrent I/O connections. ESCON supports only one I/O connection at any one time. (These are logical connections, not links.)
- Increased distance: With FICON, the distance from the channel to the ESS, or channel to switch, or switch to ESS link is increased. The distance for ESCON of 3 km is increased to up to 10 km (20 km with RPQ) for FICON channels using long-wavelength lasers with no repeaters.
- Increased link bandwidth: FICON has up to 10 times the link bandwidth of ESCON (1 GBps full duplex, compared to 200 MBps half duplex). FICON has up to more than four times the effective channel bandwidth (70 MBps compared to 17 MBps for ESCON). The FICON adapter on the ESS will also transmit/receive at 2 GB if the switch/director supports 2 GB links.
- No data rate droop effect: For ESCON channels, the droop effect started at 9 km. For FICON, there is no droop effect even at a distance of 100 km.
- Increased channel device-address support, from 1,024 devices for an ESCON channel to up 16,384 for a FICON channel.

- Greater exploitation of Parallel Access Volume (PAV): FICON allows for greater exploitation of PAV in that more I/O operations can be started for a group of channel paths.
- Greater exploitation of I/O Priority Queuing: FICON channels use frame and Information Unit (IU) multiplexing control to provide greater exploitation of the I/O Priority Queuing mechanisms within the FICON-capable ESS.
- Better utilization of the links: Frame multiplexing support on FICON channels, switches, and FICON control units such as the ESS provides better utilization of the links.

These improvements can be realized in more powerful and simpler configurations with increased throughput. The z/OS user will notice improvements over ESCON channels, with reduced bottlenecks from the I/O path, allowing the maximum control unit I/O concurrency exploitation:

- IOSQ time (UCB busy) can be reduced by configuring more alias device addresses, using PAVs. This is possible because FICON channels can address up to 16,384 devices (ESCON channels address up to 1,024; the ESS has a maximum of 4,096 devices).
- Pending time can also be reduced:
  - Channel busy conditions are reduced by FICON channel's multiple starts.
  - Port busy conditions are eliminated by FICON switches' frame multiplexing.
  - Control unit busy conditions are reduced by FICON adapters' multiple starts.
  - Device-busy conditions are also reduced, because the multiple concurrent I/O operations capability in FICON can improve the Multiple Allegiance (MA) function exploitation.

FICON channels enable a higher I/O throughput using fewer resources. The FICON architecture capability to execute multiple concurrent I/O operations, which can be sustained by the FICON link bandwidth, allows for:

- A single FICON channel can have I/O operations to multiple logical control units at the same time, by using the FICON protocol frame multiplexing.
- FICON CCW and data prefetching and pipelining and protocol frame multiplexing also allows multiple I/O operations to the same logical control unit. As a result, multiple I/O operations can be done concurrently to any logical control unit. By using the IBM ESS PAV function, multiple I/O operations are possible even to the same volume.

For a detailed description of the ESS 800, refer to *IBM TotalStorage Enterprise Storage Server Model 800*, SG24-6424.

A

# PeopleSoft data sharing configuration (option 1): customer example

In this appendix we describe a PeopleSoft data sharing configuration (option 1) implementation at a customer site.

We describe the following:

- Customer configuration
- Why data sharing?
- Designing PeopleSoft in a data sharing environment
- Partitioning schemes
- Data sharing issues

# Customer configuration

The customer has this two-way data sharing configuration, shown in Figure 7-3:

- zSeries 2064-1c4 models, each with an integrated Coupling Facility
- ESS Parallel Access Volume disks - mirrored across the members using PPRC
- DB2 for z/OS V7 two-way data sharing using dynamic SQL cashing, EDM, and buffer pools in data spaces
- Application servers (PeopleSoft V7.5 Financials and HR) on NT
- DB2 Connect V7 on NT set up with connection pooling. Half of the gateways point to member 1 and the other half point to member 2.

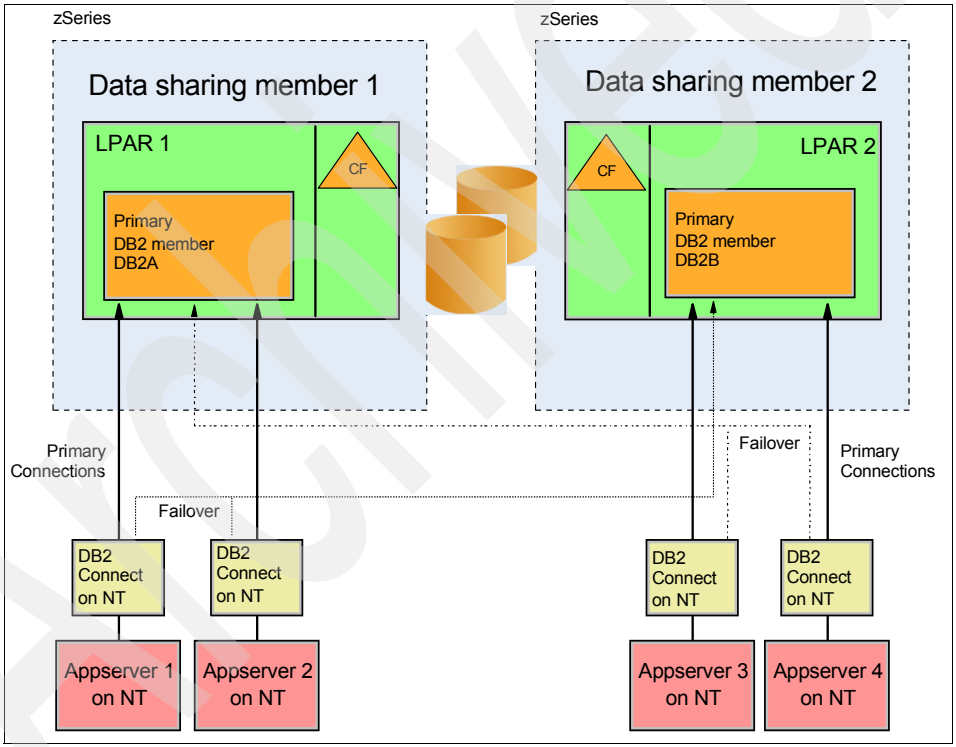



*Figure 7-3 Customer data sharing configuration*

PeopleSoft is implemented in an existing data sharing environment.

Their data sharing environment is very thoroughly tested. They have a special system contract on this purpose. They test a complete failover in production twice a year. During the first test, they move their entire workload to one member

for a week, and then during the second test, they move their workload the other direction for a week. Their contract enables them to temporarily increase capacity on the machine with the increased workload. Both members are considered as primary, and they share the workload.

## Why data sharing?

The main reason for choosing a data sharing configuration was to avoid single point of failure. The customer wanted a cost-effective alternative to its offsite disaster recovery contracts, and their best-case scenario was to recover within 48 hours to a point in time several hours prior to the disaster. They already had two data centers and two mainframes, so they consolidated their equipment in a Parallel Sysplex. Thus, they no longer have any single points of failure, and they can lose one system due to a disaster, and continue running with no loss of data on the other system.

Prior to data sharing, they were minimally satisfying their availability requirements. They had users all over the world, so they could not afford to have traditional “batch windows.” By moving to data sharing, they substantially decreased planned and unplanned outages. They have been monitoring their PeopleSoft availability for 15 months, and have been able to consistently maintain availability of > 99.9%. They had only one unplanned outage in that time frame, which was caused by human error.

### *Planned outages*

They use rolling IPLs when possible, so that even if one DB2 subsystem is shut down, the other DB2 subsystem is still available. Their planned outages are during non-peak workloads, so running their entire workload on a single DB2 is not an issue.

### *Unplanned outages*

On the rare occasion that DB2 fails, the appservers attached to that subsystem automatically fail over to the other subsystem. The entire workload runs on one subsystem for a short time until automation restarts the failed DB2. Running for this very short period of time with degraded performance is considered to be acceptable.

## Designing PeopleSoft in a data sharing environment

The customer found that their data sharing environment benefits from traditional PeopleSoft/DB2 tuning. For example, you might partition some tablespaces so that you can eliminate contention for two concurrent jobs. In a data sharing

environment, the two concurrently running jobs may be running on different LPARs. You can reduce global locking between the two jobs with partitioning, exactly the same strategy that you would use in a non-data sharing environment.

The customer followed guidelines from IBM on the setup of Workload Manager in goal mode. These settings have been instrumental in meeting Service Level Agreements. They also used neutral job classes to let WLM determine where each batch job would run.

They first tried running HR on one subsystem and Finance on the other subsystem to prevent global lock contention. However, they found that their workloads peak at different times of the day, week, and month between the various PeopleSoft modules. It is difficult to balance the workload with this type of affinity routing strategy, and they have not found it to be necessary.

They have several PeopleSoft application servers running against each subsystem. Their workload is split between them, so that approximately half of their workload goes to each subsystem. If a connection to DB2 is lost for any reason, that appserver fails over to the other subsystem.

All of their batch jobs are scheduled through their mainframe scheduler, not through the PeopleSoft process scheduler. The JCL is submitted with a neutral jobclass, so Workload Manager (WLM) determines where the job physically would run.

The customer implemented PeopleSoft in an existing DB2 data sharing group. If they were to start over, they would isolate PeopleSoft in its own DB2 group because PeopleSoft typically requires fairly new versions of software, PTFs, and so on. Their older applications are looking for stability and would prefer to be slightly behind the curve for stability reasons. Isolating PeopleSoft also gives them the opportunity to tune that DB2 specifically for PeopleSoft.

## Partitioning schemes

Their partitioning is very similar to most PeopleSoft customers. In their HR environment, they partitioned the following tables on Company and Paygroup:

```
PS_PAY_CAL_BAL_ID
PS_PAY_CHECK
PS_PAY_DEDUCTION
PS_PAY_DISTRIBUTION
PS_PAY_EARNINGS
PS_PAY_GARNISH
PS_PAY_LINE
PS_PAY_OTH_EARNS
PS_PAY_PAGE
```

```
PS_PAY_SPCL_EARNS
PS_PAY_TAX
```

In their Financials environment, they had more challenging concurrency requirements. They partitioned the following tables in that environment:

```
PS_LEDGER ' ACCOUNTING_PERIOD
PS_JRNL_LN ' BUSINESS_UNIT, JOURNAL_ID
PS_RT_RATE_TBL ' FROM_CUR
PS_JRNL_LN_TMP ' BUSINESS_UNIT, JOURNAL_ID
PS_DEPRECIATION ' BOOK, CATEGORY, DEPTID
PS_ASSET_NBV_TBL ' BOOK
```

Their partitioning schemes were designed based on their data. Other companies with different requirements and data distribution will require very different partitioning schemes. They also partitioned tables created for modifications specific to their environment.

In Financials, they also used PeopleSoft's non-shared temporary tables extensively. This has been critical to getting the concurrency required in their environment. They also made modifications to their allocations processes so that they can run concurrent allocations processes.

## Data sharing issues

The only real issue that they have seen from implementing data sharing is that they have two of everything. For example, if they change the size of a buffer pool, they have to make the change twice. It is also a little bit harder to monitor threads: When a process executes, it may run on either system, so they have to watch both. If they try to find a particular thread, they may have to execute the **-DISPLAY THREAD** command twice.

B

# Test program to validate load balancing

```
/**
** SOURCE FILE NAME: concentrator.c
**
** This sample tries to capture load balancing between sysplex members
** when connection concentrator is turned on.
** We used in our test environment a VIPA set up in round robin fashion and
** used default WLM settings. You must be careful with your settings otherwise
** you may not be able to detect load balancing behavior in your environment
** due to various factors such as WLM policy, VIPA behavior and individual
** load on each sysplex members.
**
** INSTRUCTIONS:
** 1. Copy this source program to your sqllib/samples/cli directory
** 2. Compile using following command
**    $ bldapp concentrator
** 3. Make sure you set value of MAX_CONNECTIONS > MAXAGENTS in your
**    DB2 Connect DBM configuration
** 3. Execute program as follows,
**    $ Restart db2 connect EE server (Always perform this step before running
**      this sample program)
**    $ concentrator <dbname> <user> <pswd> <numconnections> <numiterations>
**    Where dbname = Name of cataloged database
**          user   = Valid userid
**          pswd   = Valid password
```

```
**    numconnections  = Number of connections to spawn in parallel, set
**      this value equal to number of SYSPLEX members participating in
**      your data sharing environment.
**    maxiterations  = Number of select iterations to be performed by
**      each connection. (Limit this value in range 10-50)
**
** SAMPLE OUTPUT:
**
** THIS SAMPLE SHOWS LOAD BALANCING BETWEEN SYSPLEX MEMBERS WHEN CONNECTION
** CONCENTRATOR IS TURNED ON
**
** This test may not be able to detect load balancing behaviour in your
** environment due to various factors such as WLM policy, VIPA behaviour
** and individual load on each sysplex members
** Spawning 2 processes
**
** Connecting to db0g...
** Connecting to db0g...
** Connected to db0g.
** Connected to db0g.
** Process 0 executing on DB1G
** Process 1 executing on DB1G
** Process 0 executing on DB1G
** Process 1 executing on DB2G
** Process 0 executing on DB2G
** Disconnecting from db0g...
** Disconnected from db0g.
** Disconnecting from db0g...
** Disconnected from db0g.
** Load balancing worked for process 0
** Load balancing worked for process 1
** Load balancing with DB2 Connection concentrator is working in
** your environment
*****************************************************************************/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlcli1.h>
#include <time.h>
#include <unistd.h>

#include <sys/types.h>
#include <sys/wait.h>
#include <sys/ipc.h>
#include <sys/msg.h>

#include <stdio.h>
#include <string.h>
```

```
#include <stdlib.h>
#include <sqlcli1.h>
#include "utilcli.h" /* Header file for CLI sample code */

/* MAX number of connections to spawn in parallel */
#define MAXCONNECTIONS 20

#define MAXITERATIONS 50

/* Message Queue key */
#define MSGKEY 80

/* Message format */
struct msgform
{
    long mtype;
    pid_t pid;
    int concentratorWorking;
};

int main(int argc, char *argv[])
{
  char dbAlias[SQL_MAX_DSN_LENGTH + 1];
  char user[MAX_UID_LENGTH + 1];
  char pswd[MAX_PWD_LENGTH + 1];
  int rc = 0;
  int processNum = 0;
  pid_t pid[MAXCONNECTIONS];
  int concentratorWorked = FALSE;

  struct msgform rmsg; /* Used to receive message */
  int status = 0;
  int msgid = 0;

  /* number of connections to spawn in parallel */
  int numConnections = 0;

  /* number of iterations performed by each connection */
  int maxIterations = 0;

  /* check the command line arguments */
  rc = parseArgs(argc, argv, dbAlias, user, pswd, &numConnections,
&maxIterations);
  if (rc != 0)
  {
    return rc;
  }
```

```
  printf("\nTHIS SAMPLE SHOWS LOAD BALANCING BETWEEN SYSPLEX MEMBERS WHEN
CONNECTION CONCENTRATOR IS TURNED ON\n");
  printf("\nThis test may not be able to detect load balancing behaviour in
your enviroment due to various factors such as WLM policy, VIPA behaviour and
individual load on each sysplex members\n\n");


  /* Create message queue */
  msgid = msgget(MSGKEY, 0777|IPC_CREAT);

  printf("\nSpawning %d processes\n", numConnections);

  for (processNum=0; processNum< numConnections; processNum++)
  {
      if ((pid[processNum] = fork()) == 0)
        runSelectProcess(processNum, msgid, dbAlias, user, pswd,
maxIterations);
  }

  /* wait for child processes to complete */
  for (processNum=0; processNum< numConnections; processNum++)
  {
    waitpid( pid[processNum], &status, 0 );
  }

  /* Get messages from each process */
  for (processNum=0; processNum<numConnections; processNum++)
  {
    msgrcv(msgid, &rmsg, sizeof(struct msgform), 1, 0);
    if (rmsg.concentratorWorking == TRUE)
    {
      printf("Load balancing worked for process %d\n", processNum);
      concentratorWorked = TRUE;
    }
    else
      printf("Load balancing did not worked for process %d\n", processNum);
  }

  if (concentratorWorked == TRUE)
    printf("\n\nLoad balancing with DB2 Connection concentrator is working in
your environment\n");
  else
    printf("\n\nLoad balancing with DB2 Connection concentrator did not work in
your environment. Make sure you have turned on DB2 connection concentrator by
setting value of MAX_CONNECTIONS > MAXAGENTS in your DB2 Connect DBM
configuration\n");

  exit(0);
} /* main */
```

```
/* perform a basic SELECT current member FROM sysibm.sysdummy1 query for each
   process for maximum of maxIterations times
*/
int runSelectProcess(int processNum, int msgid, char dbAlias[], char user[],
char pswd[], int maxIterations)
{
  SQLHANDLE henv; /* environment handle */
  SQLHANDLE hdbc; /* connection handle */
  SQLRETURN cliRC = SQL_SUCCESS;
  int rc = 0;
  int iteration = 0;
  SQLHANDLE hstmt; /* statement handle */
  /* SQL SELECT statement to be executed */
  SQLCHAR *stmt = (SQLCHAR *)"SELECT current member FROM sysibm.sysdummy1";

  char memberName[30];
  char firstMember[30];

  struct
  {
    SQLINTEGER ind;
    SQLCHAR val[15];
  } member; /* variable to get data from the member column */


  struct msgform smsg; /* Used for sending message */

  /* Initialize message structure */
  smsg.mtype = 1;
  smsg.pid = getpid();
  smsg.concentratorWorking = FALSE;

  /* initialize the CLI application by calling a helper
     utility function defined in utilcli.c */
  rc = CLIAppInit(dbAlias,
                  user,
                  pswd,
                  &henv,
                  &hdbc,
                  (SQLPOINTER)SQL_AUTOCOMMIT_OFF);
  if (rc != 0)
  {
    return rc;
  }

  iteration = 0;
  do
  {
```

```
/* set AUTOCOMMIT OFF */
cliRC = SQLSetConnectAttr(hdbc,
                        SQL_ATTR_AUTOCOMMIT,
                        (SQLPOINTER)SQL_AUTOCOMMIT_OFF,
                        SQL_NTS);
DBC_HANDLE_CHECK(hdbc, cliRC);

/* allocate a statement handle */
cliRC = SQLAllocHandle(SQL_HANDLE_STMT, hdbc, &hstmt);
DBC_HANDLE_CHECK(hdbc, cliRC);

/* directly execute the statement */
cliRC = SQLExecDirect(hstmt, stmt, SQL_NTS);
STMT_HANDLE_CHECK(hstmt, hdbc, cliRC);

/* fetch each row, and display */
cliRC = SQLFetch(hstmt);
STMT_HANDLE_CHECK(hstmt, hdbc, cliRC);

if (cliRC == SQL_NO_DATA_FOUND)
{
  printf("\n  Data not found.\n");
}
else
{
  /* use SQLGetData to get the results */
  /* get data from column 1 */
  cliRC = SQLGetData(hstmt,
                     1,
                     SQL_C_CHAR,
                     member.val,
                     15,
                     &member.ind);
  STMT_HANDLE_CHECK(hstmt, hdbc, cliRC);


  sprintf(memberName, "%s", member.val);
  printf("Process %d executing on %s \n", processNum, memberName);
}

/* free the statement handle */
cliRC = SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
STMT_HANDLE_CHECK(hstmt, hdbc, cliRC);

/* Commit transation */
cliRC = SQLEndTran(SQL_HANDLE_DBC, hdbc, SQL_COMMIT);
rc = HandleInfoPrint(SQL_HANDLE_DBC, hdbc, cliRC, __LINE__, __FILE__);
if (rc != 0)
  printf("  The transaction failed.\n");
```

```
    if (iteration == 0)
      strcpy(firstMember, memberName);
    else
    {
      if (strcmp(firstMember, memberName) != 0)
      {
        smsg.concentratorWorking = TRUE;
        break;
      }
    }

    iteration++;
    sleep(3);
  } while (iteration < maxIterations);

  /* terminate the CLI application by calling a helper
     utility function defined in utilcli.c */
  rc = CLIAppTerm(&henv, &hdbc, dbAlias);

  msgsnd(msgid, &smsg, sizeof(struct msgform), 0);

  exit(0);
} /* runSelectProcess */

/* check command line arguments */
int parseArgs(int argc,
                    char *argv[],
                    char dbAlias[],
                    char user[],
                    char pswd[],
                    int  *numConn,
                    int  *maxIter)
{
  if (argc < 5)
  {
    printf("\nUSAGE: %s dbAlias userid  passwd numberOfConnections
numberOfIterationsForEachConnection \n", argv[0]);
    return 1;
  }
  strcpy(dbAlias, argv[1]);
  strcpy(user, argv[2]);
  strcpy(pswd, argv[3]);
  *numConn = atoi(argv[4]);
  *maxIter = atoi(argv[5]);
  if (*numConn > MAXCONNECTIONS)
  {
    printf("\n number of connections can not exceed %d\n", MAXCONNECTIONS);
    return 1;
```

```
}
if (*maxIter > MAXITERATIONS)
{
  printf("\n number of iterations can not exceed %d\n", MAXITERATIONS);
  return 1;
}

return 0;
```

C

# FlashCopy using Mainstar MS/VCR

The following text is the actual JCL used on this project. It is expected that the reader with appropriate modifications could use this JCL to obtain similar results.

1. BCSCLEAN: This job is not needed on the first run, but only if the copy step is retried. The critical DDs are JOURNAL and SYSIN. JOURNAL points to the Mainstar Journal dataset, which the Mainstar job BCSCLEAN uses to complete its cleanup. Sample JCL follows:

```
//EPBCLEAN JOB ,'VCR BCSCLEAN',CLASS=Z,MSGCLASS=X
//STEP1 EXEC PGM=VCR00010
//STEPLIB DD DSN=SYS5.MAINSTAR.V1R200G.LOAD,DISP=SHR
//VCRINI DD DSN=EP.VCR.PARMLIB(VCRINI),DISP=SHR
//BCSRECS DD DSN=EP.VCR.WRK.BCSRECS,DISP=OLD
//JOURNAL DD DSN=EP.VCR.JRNL,DISP=OLD
//VCRPRINT DD SYSOUT=*
//SYSIN DD *
   BCSCLEAN
   JOURNAL(JOURNAL)
```

2. STOP SOURCE DB2: Use this DB2 command to stop the source DB2 subsystem very briefly (about 3 minutes). Stopping DB2 is highly recommended. Otherwise, the target subsystem may have in-flight threads and will not come up smoothly. Note that we only stopped the Payroll

databases because there were no other active databases during FlashCopy, but if there were other active databases we recommend they all be stopped.

```
-STOP DB (PAYROLL)
```

3. FLASHCOPY: This job copies source volumes to target volumes extremely quickly, usually in several minutes. The critical DDs are JOURNAL and SYSIN. SYSIN identifies the maximum volumes to copy in a task (COPYCMDLIMIT) and the number of tasks (DSSTASKS). The copy process is so quick, there was little advantage to altering these parameters. SYSIN also identifies the source volumes and target volumes. Notice that the clause `FROM-VOLSER` uses a mask of `SRC*` to copy from all volumes with a VOLSER beginning with SRC. SRC* covers 128 volumes, and the remaining five volumes are individually listed starting with `SP4A40`.

   The `TO-VOLSER` clause identifies which volumes received the data. Notice the use of the `TGT*` mask: The clause USERCATALOGS identifies how to copy the DB2 catalog datasets.

   Because this job creates the important journal file, it is best to have the first step an IDCAMS delete of the journal file.

   When this job is done, the source database can be restarted.

```
//EPCP24X4 JOB ,'VCR COPY',CLASS=Z,MSGCLASS=X
//STEP0 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DEL EP.VCR.JRNL
   DEL EP.VCR.WRK.UCATBKUP.*
   SET MAXCC=0
//STEP1 EXEC PGM=VCR00010
//STEPLIB DD DSN=SYS5.MAINSTAR.V1R200G.LOAD,DISP=SHR
//VCRINI DD DSN=EP.VCR.PARMLIB(VCRINI),DISP=SHR
//BCSRECS DD DSN=EP.VCR.WRK.BCSRECS,DISP=OLD
//JOURNAL DD DSN=EP.VCR.JRNL,DISP=(,CATALG,),
// RECORG=KS,KEYLEN=64,KEYOFF=0,UNIT=SYSDA,
// LRECL=600,SPACE=(CYL,(10,10))
//VCRPRINT DD SYSOUT=*
//SYSIN DD
   COPY
      DATA-MOVER (COPYCMDLIMIT(24)
      DSSTASKS(4))
      FROM-VOLSER (SRC*
      SP4A40,SP4A41,SP4B40,SP4B41,SP4C40)
      TO-VOLSER (TGT*
      SP4A42,SP4A43,SP4B42,SP4B43,SP4C31)
      USERCATALOGS(
      CATALOG.DB1M CATALOG.DB1X
      CATALOG.DS1M CATALOG.DS1X )
      JOURNAL-DDN(JOURNAL)
```

**Note:** It is recommended that the DSSTASKS parameter equal the number of LSSs containing source data. We recorded no significant benefit when we increased the DSSTASKS parameter greater than four.

4. RESTART SOURCE DB2: As soon as the brief Copy job completes, the source DB2 can be brought back online. As indicated in the timeline below, this is just several minutes.

```
-START DB(PAYROLL)
```

5. COPYCHECK: This job is optional. It monitors the progress of the ESS background copy. It is important to note that after the job completes (see step 3 on page 126), the source database is available.

```
//EPCOPYCK JOB ,'VCR COPYCHECK',CLASS=Z,MSGCLASS=X
//STEP1 EXEC PGM=VCR00010
//STEPLIB DD DSN=SYS5.MAINSTAR.V1R200G.LOAD,DISP=SHR
//VCRINI DD DSN=EP.VCR.PARMLIB(VCRINI),DISP=SHR
//JOURNAL DD DSN=EP.VCR.JRNL,DISP=(,CATALG,),DISP=SHR
//VCRPRINT DD SYSOUT=*
//SYSIN DD
   COPYCHECK
   WAIT(12)
   JOURNAL-DDN(JOURNAL)
```

6. RENAME: Critical job for the target DB2 subsystem, as this renames the tablespace High Level Qualifier (HLQ).

```
//EPRNAME JOB ,'VCR COPY',CLASS=Z,MSGCLASS=X
//STEP0 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
   DEL EP.VCR.WRK.BCSRECS
   DEL EP.VCR.WRK.IDCAMS
   DEL EP.VCR.WRK.VOLBKUP
/*
//STEP1 EXEC PGM=VCR00010
//STEPLIB DD DSN=SYS5.MAINSTAR.V1R200G.LOAD,DISP=SHR
//VCRINI DD DSN=EP.VCR.PARMLIB(VCRINI),DISP=SHR
//SORTWRK01 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//SORTWRK02 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//SORTWRK03 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//SORTWRK04 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//SORTWRK05 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//SORTWRK06 DD UNIT=SYSDA,SPACE=(CYL(10,10))
//VCRPRINT DD SYSOUT=*
//DRSTATS DD SYSOUT=*
//MSPRINT DD SYSOUT=*
//JOURNAL DD DSN=EP.VCR.JRNL,DISP=OLD
```

```
//BCSRECS DD DSN=EP.VCR.WRK.BCSRECS,DISP=(,CATLG,)
//MSWORK DD DSN=EP.VCR.WRK.IDCAMS,DISP=(,CATLG,)
//VOLBKUP DD DSN=EP.VCR.WRK.VOLBKUP,DISP=(,CATLG,)
//SYSIN DD *
   RENAME
   SPEED
      MAX-TASKS(9)
      JOURNAL-DDN(JOURNAL)
      RECATALOG(N)
      DATACLASS(SOURCE)
      MGMTCLAS(SOURCE)
      STORCLAS(SOURCE)
      RENAME-MASKS (
         DB1M.*.* DB1X.*.*
         DS1M.*.* DS1M.*.*)
```

7. DB2UPDATE: This critical job updates the BSDS on the target subsystem. Note the source and target DB2 pairing on the `DB2UPDATE` clause.

```
//EPDB2UPD JOB ,'VCR DB2 UPDATE',CLASS=Z,MSGCLASS=X
//STEP1 EXEC PGM=VCR00010
//STEPLIB DD DSN=SYS5.MAINSTAR.V1R200G.LOAD,DISP=SHR
//VCRINI DD DSN=EP.VCR.PARMLIB(VCRINI),DISP=SHR
//BSDS01 DD DSN=DS1X.BSDS01, DISP=SHR
//BSDS02 DD DSN=DS1X.BSDS02,DISP=SHR
//DBD01 DD DISP=OLD
// DSN=DS1X.DSNDBC.DSNDB01.DBD01.I0001.A001
//SYSIN DD *
   DB2UPDATE
      DB2-HLQS(DS1M DS1X DB1M DB1M)
      JOURNAL-DDN(JOURNAL)
```

8. DSNJU003: This optional job is needed if you are using DDF. It updates the BSDSs with the new communication information. The critical DD is SYSIN: Its values vary for each installation. These are the values we used.

```
DDF LOCATION=DB2DS1X.LUNAME=DB2APL1X,PORT=5118,RESPORT=5119,PAMSVCRSSWORD=
```

9. START TARGET DB2 SUBSYSTEM

# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 129. Note that some of the documents referenced here may be available in softcopy only.

- *IBM TotalStorage Enterprise Storage Server Model 800*, SG24-6424

## Other publications

These publications are also relevant as further information sources:

- *IBM DB2 Connect User's Guide, Version 8*, SC09-4835

## Online resources

These Web sites are also relevant as further information sources:

- White paper about data cloning process using FlashCopy and Mainstar with a PeopleSoft Payroll database

  http://www-1.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100334

- White paper *Dynamic VIPA routing and workload balancing in a DB2 data sharing environment*

  http://www-1.ibm.com/support/docview.wss?rs=64&uid=swg27000593

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications, and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

**ibm.com**/redbooks

# Help from IBM

IBM Support and downloads

**ibm.com**/support

IBM Global Services

**ibm.com**/services

# Index

IBM Redbooks

PeopleSoft V8 on zSeries Using Sysplex Data Sharing and Enterprise Storage Systems

(0.2”spine)
0.17”<->0.473”
90<->249 pages

# PeopleSoft V8 on zSeries Using Sysplex Data Sharing and Enterprise Storage Systems

**zSeries sysplex data sharing and PeopleSoft**

**DB2 Connect and load balancing**

**ESS performance with PeopleSoft**

This IBM Redbook describes the implementation of PeopleSoft Version 8.4 on a zSeries Parallel Sysplex data sharing environment. It discusses the benefits of running PeopleSoft in a sysplex with data sharing, and explains the performance considerations and issues when implementing PeopleSoft in a data sharing environment.

The book also investigates the load balancing of PeopleSoft workloads using DB2 Connect to connect to the sysplex DB2 data sharing members. It describes functional and volume tests of load balancing, and a section about Enterprise Storage Systems describes PeopleSoft performance benefits when using ESS.

**INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION**

**BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**
**ibm.com**/redbooks

SG24-6093-00 ISBN 0738498734