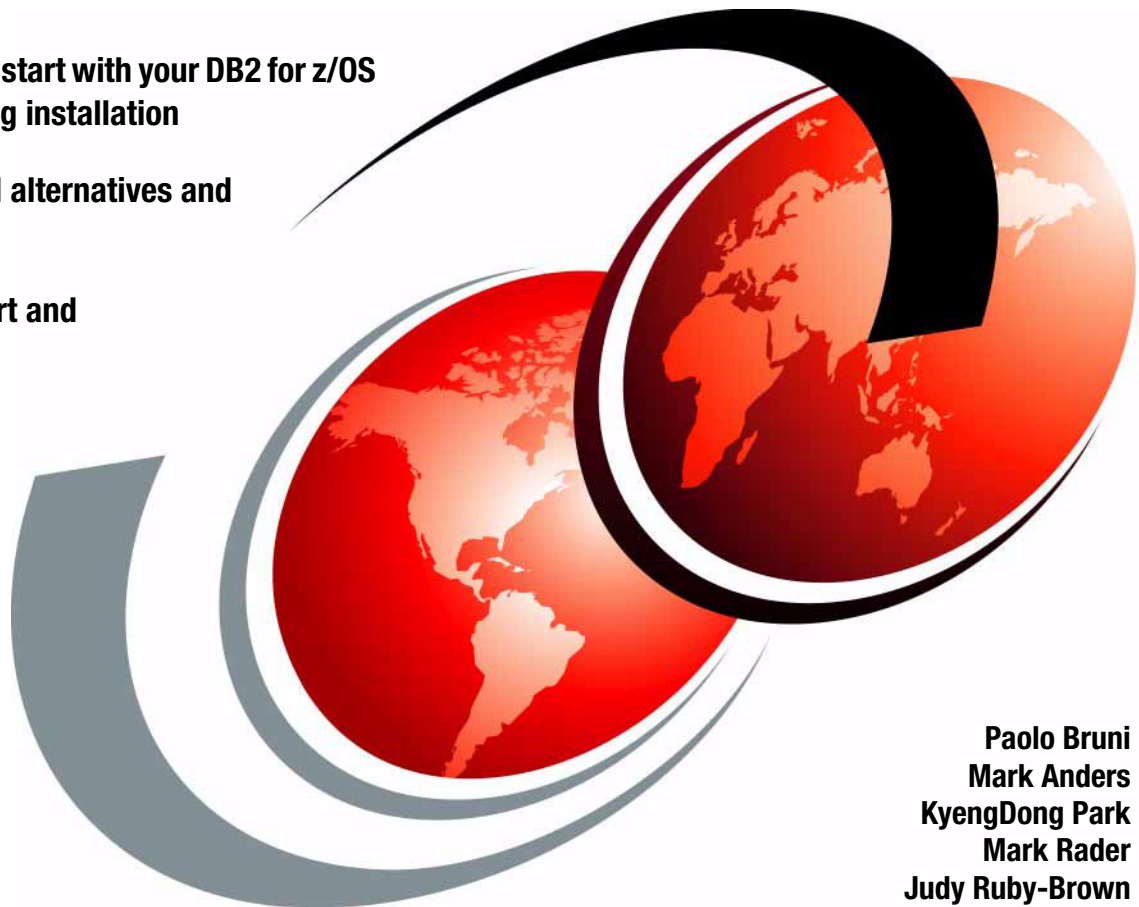


# DB2 for z/OS: Data Sharing in a Nutshell

Get a quick start with your DB2 for z/OS  
data sharing installation

Understand alternatives and  
priorities

Find support and  
references



Paolo Bruni  
Mark Anders  
KyengDong Park  
Mark Rader  
Judy Ruby-Brown





International Technical Support Organization

**DB2 for z/OS:  
Data Sharing in a Nutshell**

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

**First Edition (October 2006)**

This edition applies to IBM DB2 Version 9.1 for z/OS (program number 5635-DB2) and earlier currently supported releases.

**© Copyright International Business Machines Corporation 2006. All rights reserved.**

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

# Contents

<b>Notices</b> .....	vii
Trademarks .....	viii
<b>Preface</b> .....	ix
Our assumptions .....	ix
The team that wrote this redbook .....	x
Become a published author .....	xii
Comments welcome .....	xiii
<b>Chapter 1. Introduction</b> .....	1
1.1 Why should you implement DB2 data sharing? .....	2
1.2 Overview of Parallel Sysplex and DB2 data sharing .....	2
1.3 Business value of data sharing .....	5
1.3.1 Improved data availability .....	6
1.3.2 Extended processing capacity .....	7
1.3.3 Configuration flexibility .....	9
1.3.4 Higher transaction rates .....	10
1.3.5 Application interface unchanged .....	11
1.4 Roadmap to implementing data sharing .....	11
<b>Chapter 2. Data sharing architecture</b> .....	15
2.1 Parallel database architecture alternatives .....	16
2.2 Data sharing design for scalability .....	18
2.2.1 Global locking and the lock structure .....	19
2.2.2 Managing changed data and group buffer pools .....	21
2.3 Data sharing design for continuous availability .....	25
2.4 Configuration flexibility and systems management .....	31
2.5 Application considerations in data sharing .....	35
2.5.1 Portability .....	36
2.5.2 Commit and lock avoidance .....	36
2.5.3 Concurrency .....	37
<b>Chapter 3. The coupling facility</b> .....	39
3.1 Structure sizing .....	40
3.1.1 Group buffer pools .....	40
3.1.2 Sizing the lock structure .....	46
3.1.3 Sizing the SCA structure .....	47
3.2 Auto Alter .....	47
3.2.1 Implementing Auto Alter .....	50

3.3 Duplexing . . . . .	53
3.3.1 Group buffer pools (GBP) or user-managed pools . . . . .	54
3.3.2 System-managed duplexing . . . . .	55
3.4 CF configuration alternatives. . . . .	56
3.4.1 ICF-only: Double failure for the lock and SCA structures . . . . .	57
3.4.2 System-managed duplexing for DB2 lock and other structures . . . . .	59
3.4.3 External CF, such as z890 or z9 BC. . . . .	59
3.5 CFRM policy . . . . .	60
3.6 Best practices for the coupling facility . . . . .	63
<b>Chapter 4. Implementing data sharing . . . . .</b>	<b>65</b>
4.1 Naming conventions . . . . .	66
4.1.1 Group name . . . . .	66
4.1.2 Group attachment name . . . . .	67
4.1.3 Subsystem identifier (SSID) . . . . .	68
4.1.4 Log data set names. . . . .	68
4.1.5 Bootstrap data set (BSDS) . . . . .	68
4.1.6 Distributed Data Facility (DDF) related parameters . . . . .	69
4.1.7 Catalog alias . . . . .	71
4.1.8 Temporary work file database. . . . .	71
4.1.9 Some more naming recommendations . . . . .	71
4.1.10 Examples of naming conventions . . . . .	72
4.1.11 Best practices for naming conventions . . . . .	73
4.2 Logging . . . . .	73
4.2.1 Active log data sets. . . . .	73
4.2.2 Archive log . . . . .	74
4.2.3 Best practices for DB2 logging . . . . .	74
4.3 DSNZPARMs . . . . .	74
4.3.1 Data sharing system parameters . . . . .	74
4.3.2 Some other important DSNZPARMs. . . . .	75
4.3.3 IRLM parameters . . . . .	76
4.3.4 Information stored in the BSDS. . . . .	77
4.4 Renaming an existing non-data sharing member . . . . .	78
4.4.1 IPL required for these tasks . . . . .	79
4.4.2 Enable data sharing tasks. . . . .	79
4.5 Enabling the data sharing group . . . . .	85
4.5.1 Additional tasks . . . . .	87
4.6 Adding the second member . . . . .	87
4.7 Removing a member. . . . .	88
<b>Chapter 5. Dynamic workload balancing . . . . .</b>	<b>89</b>
5.1 Objectives of workload balancing . . . . .	90
5.2 Workload Manager . . . . .	92

5.3 Dynamic virtual IP addressing (DVIPA) . . . . .	94
5.4 Sysplex Distributor . . . . .	95
5.5 Distributed data facility (DDF) . . . . .	95
5.6 Stored procedures . . . . .	97
5.7 Batch work . . . . .	98
5.8 WebSphere . . . . .	99
5.9 CICSplex Systems Manager (CP/SM) . . . . .	100
5.10 IMS Transaction Manager (IMS TM) . . . . .	101
5.11 Dynamic workload balancing best practices . . . . .	103
<b>Chapter 6. Operations . . . . .</b>	<b>105</b>
6.1 Recovery of DB2 objects . . . . .	106
6.1.1 Log record sequence number (LSRN) . . . . .	106
6.1.2 Group buffer pool recovery pending (GRECP) . . . . .	106
6.1.3 Logical page list (LPL) . . . . .	107
6.1.4 Recommendations for GRECP/LPL recovery . . . . .	107
6.1.5 Best practices for GRECP/LPL recovery . . . . .	109
6.2 Component failure . . . . .	109
6.2.1 DB2 subsystem failure . . . . .	110
6.2.2 z/OS system failure . . . . .	111
6.2.3 CF failure . . . . .	111
6.3 Sysplex failure management (SFM) policy . . . . .	112
6.4 Automatic restart manager (ARM) . . . . .	112
6.5 Restart Light . . . . .	112
6.5.1 Best practices for failures . . . . .	113
6.6 Commands . . . . .	114
6.6.1 Basics . . . . .	114
6.6.2 CF and structure related commands . . . . .	115
6.6.3 IRLM commands . . . . .	117
6.6.4 DB2 commands . . . . .	117
6.7 Multi-system DB2 diagnostic dumps . . . . .	118
6.8 Disaster recovery . . . . .	119
6.9 Rolling maintenance . . . . .	119
6.9.1 Service recommendations . . . . .	120
6.9.2 Best practices for maintenance . . . . .	120
<b>Chapter 7. Advanced topics . . . . .</b>	<b>121</b>
7.1 CLOSE YES and CLOSE NO table spaces . . . . .	122
7.1.1 Best practice . . . . .	123
7.2 Performance . . . . .	123
7.3 Eliminating false contention . . . . .	124
7.4 How many CF engines are necessary? . . . . .	126
7.5 Lock avoidance: CLSN versus GCLSN . . . . .	128

7.6 The problem of runaway threads. . . . .	128
7.7 Usermod for routing to multiple DB2s on a z/OS image . . . . .	130
7.7.1 Multiple DB2 members in one z/OS image . . . . .	130
7.8 Determining the number of threads. . . . .	131
7.8.1 REXX tools package . . . . .	132
7.8.2 Statistics Spreadsheet support . . . . .	133
<b>Chapter 8. Best practices</b> . . . . .	135
8.1 Table of best practice recommendations . . . . .	136
<b>Related publications</b> . . . . .	139
IBM Redbooks . . . . .	139
Other publications . . . . .	139
Online resources . . . . .	140
How to get IBM Redbooks . . . . .	142
Help from IBM . . . . .	142
<b>Index</b> . . . . .	143



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

# Trademarks

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

CICS®	ibm.com®	RMF™
CICSplex®	IMS™	Sysplex Timer®
DB2®	MQSeries®	System z™
DB2 Connect™	MVS™	System z9™
DFSMSdftp™	OMEGAMON®	Tivoli®
DRDA®	OS/390®	VTAM®
eServer™	Parallel Sysplex®	WebSphere®
ESCON®	RACF®	z/OS®
FICON®	Redbooks™	zSeries®
IBM®	Redbooks (logo)  ™	z9™

System z is the brand name to indicate both the zSeries and System z9 families of products.

The following terms are trademarks of other companies:

Java, JDBC, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

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

# Preface

DB2® for z/OS® takes advantage of data sharing technology in a z Parallel Sysplex® to provide applications with full concurrent read and write access to shared DB2 data. Data sharing allows users on multiple DB2 subsystems, members of a data sharing group, to share a single copy of the DB2 catalog, directory, and user data sets.

Data sharing provides improvements to availability and capacity without impacting existing applications.

The road to data sharing might seem arduous to a novice user, but once you have started to learn terminology and gain basic understanding, things will become much easier.

This IBM® Redbook is meant to facilitate your journey towards data sharing by providing a cookbook approach to the main tasks in enabling data sharing and workload balancing. It does not have all the answers, because it is a brief summary of a large field of knowledge, but it contains the key elements and it points you in the right direction to get more details. Throughout this document we assume that your sysplex environment is set up and a DB2 subsystem exists at a currently supported level.

## Our assumptions

We made the following assumptions in writing this book:

- ▶ You already have a production Parallel Sysplex installed, with sysplex timers, coupling facilities (CFs), CF links, and CF structures. All disks are shared in your Parallel Sysplex.
- ▶ You already have experience with DB2 in a non-data sharing environment.
- ▶ You are running DB2 for z/OS Version 8 (DB2 V8) in New Function Mode
- ▶ You are at the very early stages of planning on implementing 2-way data sharing.
- ▶ The primary reference for data sharing topics is the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.
- ▶ For the details and differences in data sharing functions between DB2 V7 and V8, refer to *DB2 UDB for z/OS Version 8: Everything You ever Wanted to Know,... and More*, SG24-6079. For DB2 V8 performance, see *DB2 for z/OS Version 8 Performance Topics*, SG24-6465.

**Important:** If you are on DB2 Version 7, you also can use this redbook. The V7 differences are as follows:

- ▶ Lock protocol 1 is V7, while lock protocol 2 is V8 NFM and above.
- ▶ GRECP/LPL recovery is not auto started by DB2 in V7.
- ▶ Restart Light does not remove as many retained locks in V7, nor does it wait for commit coordinators to be started.
- ▶ GBP default thresholds are higher in V7 than in V8. Use those in V8.
- ▶ The need for the BIND option RELEASE DEALLOCATE is less in V8 versus V7 due to lock protocol 2.
- ▶ All V7 *write and register page* and *read for castout* CF requests are on a single page basis versus CF batching in V8.
- ▶ Location aliases for distributed access to a subset of the data sharing group are not available in V7.

The most important data sharing functions introduced by DB2 Version 9.1 for z/OS are also explicitly mentioned in this redbook.

## 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.

**Paolo Bruni** is an Information Management software Project Leader at the International Technical Support Organization, San Jose Center. He has authored several IBM® Redbooks™ about DB2 for z/OS and related tools, and has conducted workshops and seminars worldwide. During Paolo's many years with IBM, both in development and in the field, his work has been mostly related to database systems.

**Mark Anders** is a Consulting IT Specialist with IBM Advanced Technical Support, Americas, located in the Dallas Systems Center. He has more than 30 years of IBM experience supporting DB2 and IMS™ as a Systems Engineer, Database Administrator, and IT Specialist. For the past 11 years, he has provided DB2 for OS/390® and z/OS technical support and consulting services with the Dallas Systems Center (now ATS). His areas of expertise include DB2 data sharing, performance, and availability.

**KyengDong Park** is a Certified IT Specialist with Global Technology Services in IBM Korea. He has almost 13 years of experience in DB2 field as a System Engineer, DBA, and IT Specialist. He has been with IBM for more than seven years, and for the past five years he worked for one of the largest customers in Korea. His areas of expertise include DB2 installation and migration, data sharing service, system performance tuning, and problem determination.

**Mark Rader** is a Consulting IT Specialist at ATS, Dallas. He has 22 years of technical experience with IBM, including large systems, communications, and databases. He has worked primarily with DB2 for MVS™, OS/390, and z/OS for more than 17 years, and has specialized in data sharing, performance, and related topics for the past nine years. He has been in the Dallas Systems Center (now ATS) for the last five years.

**Judy Ruby-Brown** is a Senior Consulting IT DB2 Specialist in the US with the Advanced Technical Support (ATS) organization. She is also an IBM Senior Certified IT Specialist. She has supported DB2 for OS/390 and z/OS for 16 years in the IBM Dallas Systems Center (now ATS). Her areas of expertise include disaster recovery, Parallel Sysplex and DB2 data sharing, high availability, and JDBC/SQLJ capabilities. She has presented on these topics at IDUG in the US, Europe, and Asia Pacific, as well as at the DB2 Technical Conferences and SHARE. She published the first successful DB2 offsite disaster recovery scenario in 1989. That scenario has been incorporated into each DB2 Administration Guide since DB2 V2R3. She holds a degree in Mathematics from the University of Oklahoma. Judy has co-authored the Redbooks SAP R/3 and DB2 for OS/390 Disaster Recovery, SG24-5343, DB2 for z/OS and OS/390: Ready for Java™, SG24-6435, and Disaster Recovery with DB2 UDB for z/OS, SG24-6370.

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

Frank Kyne  
Bob Haimowitz  
Rich Conway  
IBM ITSO, Poughkeepsie Center

Angelo Corridori  
Andrea Harris  
David Raften  
IBM Systems & Technology Group, Poughkeepsie

Michael B. Fast  
IBM Washington Systems Center, ATS

John Campbell  
Jeff Josten

Gopal Krishnan  
Roger Miller  
Chris Munson  
Mary Petras  
Akira Shibamiya  
Nigel Slinger  
John Tobler  
Chung Wu  
IBM DB2 Development, Silicon Valley Lab

Willie Favero  
Glenn McGeoch  
IBM USA

Steve Zemblowski  
Guy Shevik  
IBM USA, ATS Americas

Bryce Krohn  
Krohn Enterprises Inc.

Franco Meroni  
IBM Italy

Pierre Cassier  
IBM France, Montpellier

Carsten Rasmussen  
IBM Denmark

## 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 have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

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](http://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](http://ibm.com/redbooks)

- ▶ Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- ▶ Mail your comments to:

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







# Introduction

In this chapter we provide introductory information to data sharing. We briefly describe Parallel Sysplex, the characteristics of DB2 data sharing, and its business value. We then list the main tasks to be completed to implement data sharing.

We discuss the following topics:

- ▶ Why should you implement DB2 data sharing?
- ▶ Overview of Parallel Sysplex and DB2 data sharing
- ▶ Business value of data sharing
- ▶ Roadmap to implementing data sharing

## 1.1 Why should you implement DB2 data sharing?

Many customers from many industries have incorporated DB2 data sharing into their information technology infrastructure to address challenges that arose in supporting their business requirements.

<b>Availability</b>	Increased business access to data any time of any day demands protection from planned and unplanned outages.
<b>Capacity</b>	A single system may be a constraint to rapid business growth, but splitting the database would introduce additional problems.
<b>Growth</b>	Easily accommodating business targets means providing scalable, non-disruptive growth options.
<b>Workload balancing</b>	Efficiency requires the effective utilization of available processor capacity for mixed workloads and the ability to handle unpredictable workload spikes.
<b>Systems management</b>	Consolidation of multiple systems allows easier systems management.

If you face one or more of these challenges, DB2 data sharing can provide the foundation you need to address them.

## 1.2 Overview of Parallel Sysplex and DB2 data sharing

The data sharing function of the licensed program DB2 for z/OS enables multiple applications to read from, and write to, the same DB2 data concurrently. The applications can run on different DB2 subsystems residing on multiple central processor complexes (CPCs) in a Parallel Sysplex.

A *Sysplex* is a group of z/OS systems that communicate and cooperate with one another using specialized hardware and software. They are connected and synchronized through a Sysplex Timer® or System z™ Server Time Protocol (STP), and enterprise systems connection (ESCON®) or fiber connection (FICON®) channels. A *Parallel Sysplex* is a Sysplex that uses one or more *coupling facilities* (CFs), which provide high-speed caching, list processing, and lock processing for any applications on the Sysplex. For information on Parallel Sysplex technology and benefits, go to:

<http://www.ibm.com/systems/z/resiliency/parsys.html>

Figure 1-1 illustrates a Parallel Sysplex. A Parallel Sysplex can include CPCs of different generations (for example, an IBM zSeries® System z890 or z990 and an IBM System z9™ BC or z9 EC).

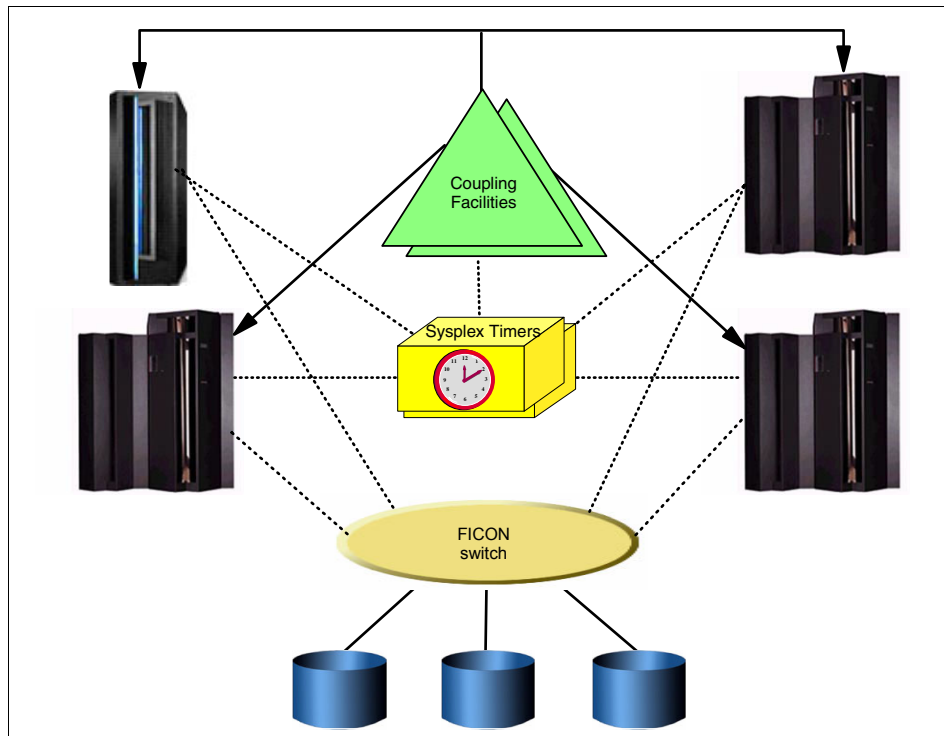


Figure 1-1 Parallel Sysplex with multiple IBM z9 processor complexes, Sysplex Timers, coupling facilities and shared disk.

A collection of one or more DB2 subsystems that share DB2 data is called a *data sharing group*. DB2 subsystems that access shared DB2 data must belong to a data sharing group. A DB2 subsystem that belongs to a data sharing group is a *member* of that group.

Each member can belong to one, and only one, data sharing group. All members of a data sharing group share the same DB2 catalog and directory, and all members must reside in the same Parallel Sysplex. Currently, the maximum number of members in a data sharing group is 32.

Figure 1-2 shows members of a DB2 data sharing group. Access to any of the members allows access to any of the data in the group.

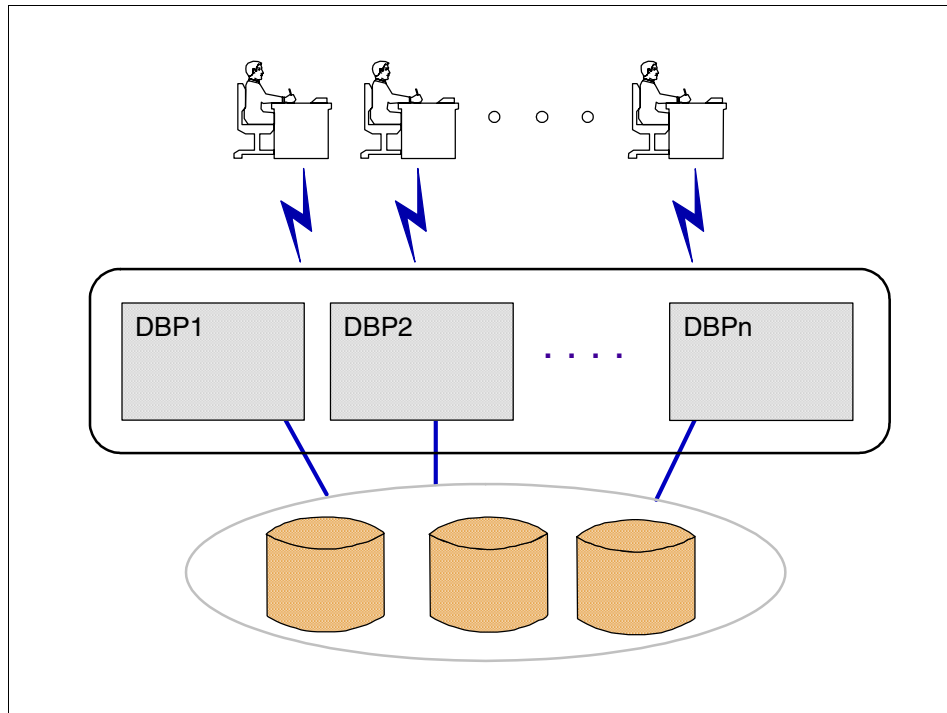
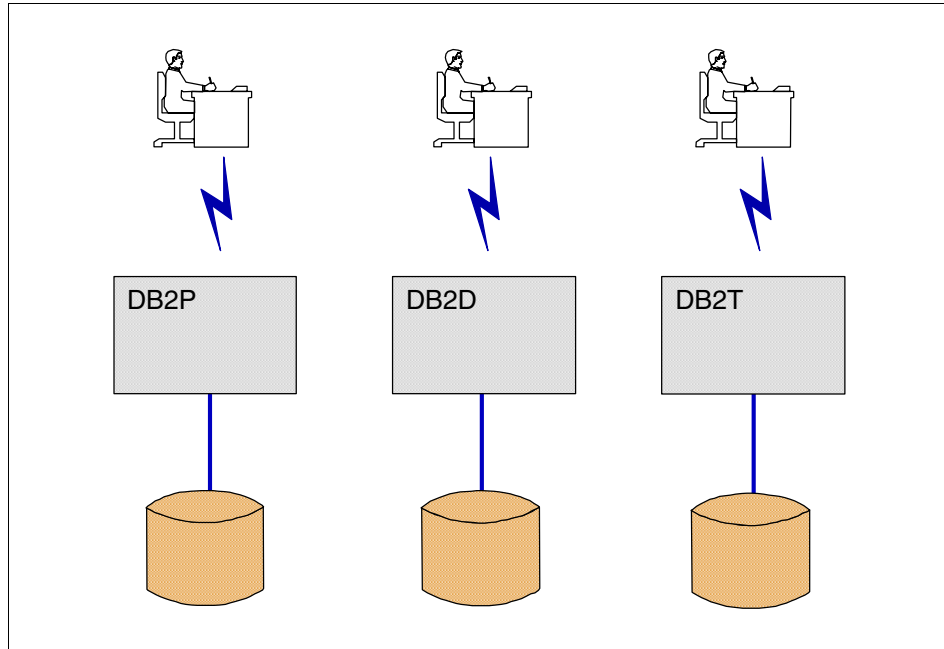


Figure 1-2 DB2 members in a data sharing group

All members of a data sharing group can read and update the same DB2 data simultaneously. Therefore, all data that different members of the group can access must reside on shared disks.

Some capabilities described in this redbook can be used in both data sharing and non-data sharing environments. In this book we use the term *data sharing environment* to describe a situation in which a data sharing group has been defined with at least one member. In a non-data sharing environment, no group is defined. Figure 1-3 shows three DB2 subsystems that are not using data sharing. Resources of each DB2 are available only to the set of data and set of applications that are defined to it.



*Figure 1-3 Separate DB2 subsystems in a non-data sharing environment.*

If you are accessing DB2P, you are limited to the data in DB2P. If DB2P is not available, you have no access to data. To access data in DB2P and DB2D at the same time, here is what you must do. Either set up separate connectivity to DB2P, and have the application combine the data (or combine the data yourself), or else ensure that there is a distributed connection between DB2P and DB2D.

## 1.3 Business value of data sharing

DB2 data sharing allows customers to provide the highest level of scalability, performance, and continuous availability to enterprise applications that use DB2 data. Using DB2 data sharing, you can:

- ▶ Improve the availability of DB2 data.
- ▶ Extend the processing capacity of your system.
- ▶ Configure your environment more flexibly.
- ▶ Increase your transaction rates.

Your investment in your current application environment is protected, because most applications do not require SQL changes to use data sharing. Future development is simpler because all data is local to all applications. In the following sections we discuss these potential benefits of data sharing.

### 1.3.1 Improved data availability

Demand is increasing for access to DB2 data — all day, every day. Data sharing helps you meet your service objectives by improving data availability during both planned and unplanned outages.

Data sharing provides multiple paths to data. This means that a member can be down, and applications can still access the data through other members of the data sharing group. As Figure 1-4 illustrates, when an outage occurs and one member is down, transaction managers can dynamically direct new application requests to another member of the group for both SNA and TCP/IP transactions.

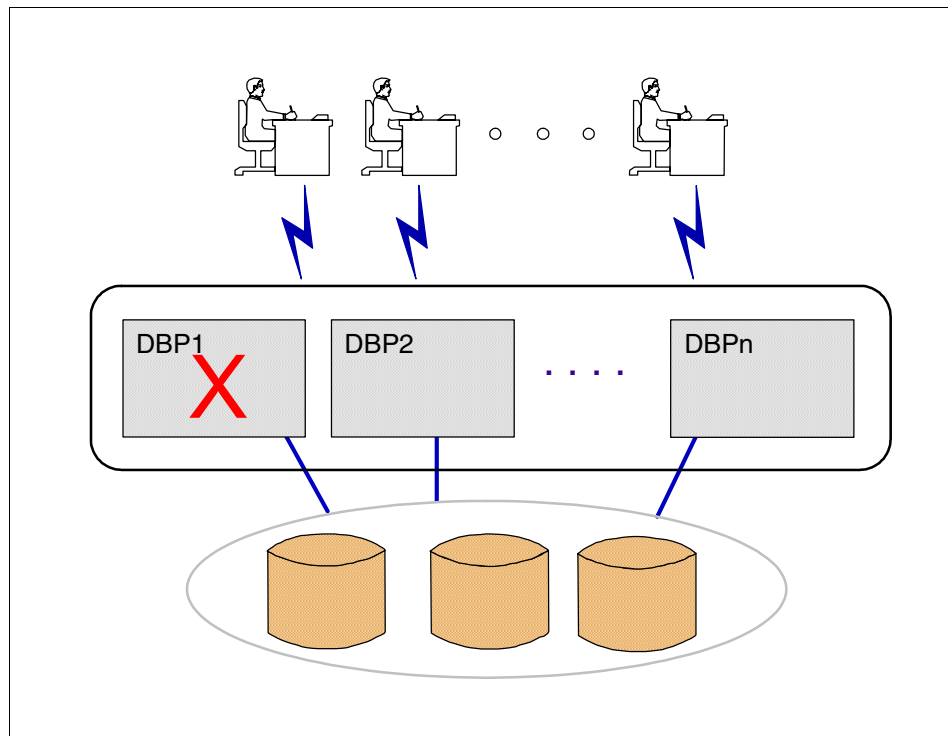


Figure 1-4 Handling outages

Each of the applications maintains availability to the data even if one DB2 member is down for maintenance or due to a failure.

While increasing data availability has some performance cost, the overhead is minimal. DB2 provides efficient locking and caching mechanisms by using coupling facility capabilities and taking advantage of the high availability features of coupling facilities and the Parallel Sysplex.

## 1.3.2 Extended processing capacity

As you move more data processing onto DB2, your processing needs can exceed the capacity of a single LPAR or a single DB2 subsystem. This section describes how data sharing meets your ever-increasing capacity needs.

### Without DB2 data sharing

Without DB2 data sharing, you have the following options for addressing increased capacity needs:

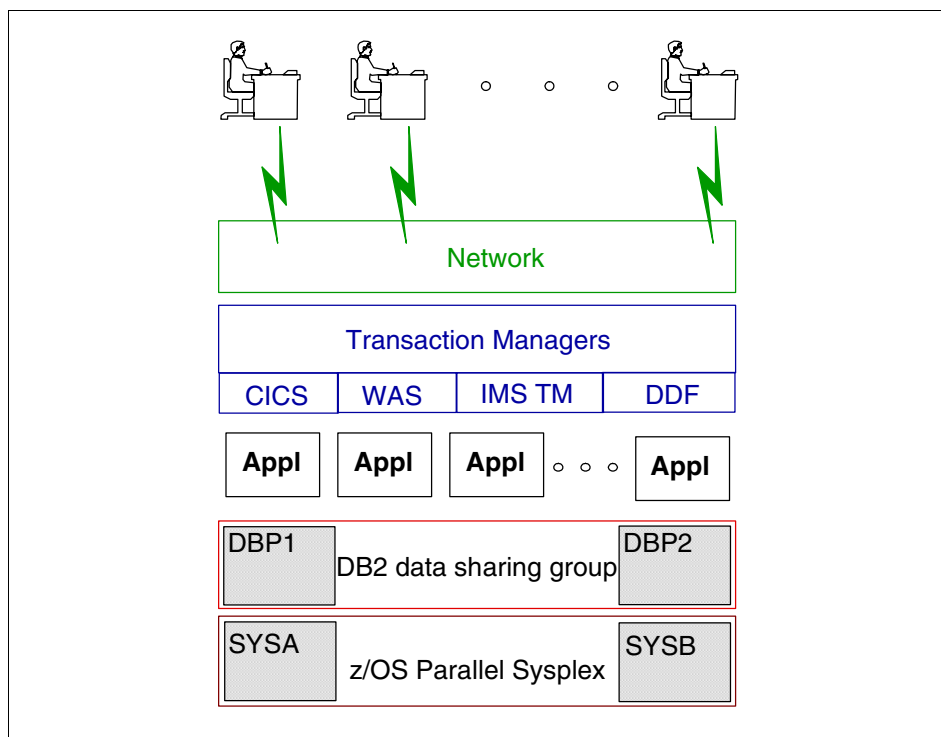
- ▶ Copy the data, or split the data between separate DB2 subsystems.  
This approach requires that you maintain separate copies of the data. There is no communication between or among DB2 subsystems, and no shared DB2 catalog or directory.
- ▶ Install another DB2 subsystem and rewrite applications to access the data as distributed data.  
This approach might relieve the workload on the original DB2 subsystem, but it requires changes to your applications and has performance overhead of its own. Nevertheless, if DB2 subsystems are separated by great distance or DB2 needs to share data with another system, the distributed data facility is still your only option.
- ▶ Install a larger processor and move the data and applications to that machine if hardware capacity is the limiting factor.  
This approach demands servers large enough for the expected workload and that your system come down while you move to the new machine.

### With DB2 data sharing

With DB2 data sharing, you get the following benefits:

- ▶ Support for incremental growth:  
A Parallel Sysplex can grow incrementally, allowing you to add processing power in granular units and in a non-disruptive manner. The coupling technology of Parallel Sysplex along with the additional CPU power results in more throughput for users' applications. You no longer need to manage multiple copies of data, and all members of the data sharing group share a single DB2 catalog and directory.
- ▶ Workload balancing:  
DB2 data sharing provides workload balancing so that when the workload increases or you add a new member to the group, you do not need to distribute your data or rewrite your applications. Applications can access the same data through the new member just as easily as through any of the existing members.

Figure 1-5 shows how the members of a data sharing group are independent of the applications, allowing non-disruptive growth to the environment.



*Figure 1-5 Members of a data sharing group are independent of the applications, supporting non-disruptive, scalable growth and workload balancing.*

Transaction managers, including DDF, work closely with the Workload Manager (WLM) component of z/OS to ensure that incoming requests are optimally balanced across the members of a Parallel Sysplex. This is possible because all DB2 members of the data sharing group have the same concurrent and direct read-write access to the data.

► Capacity when you need it:

A data sharing configuration can easily handle workload growth. Peaks within capacity can be handled by workload distribution. In case of sustained growth, you can add a new data sharing member to meet planned growth requirements without disruption.



Figure 1-6 shows a third member added for growth.

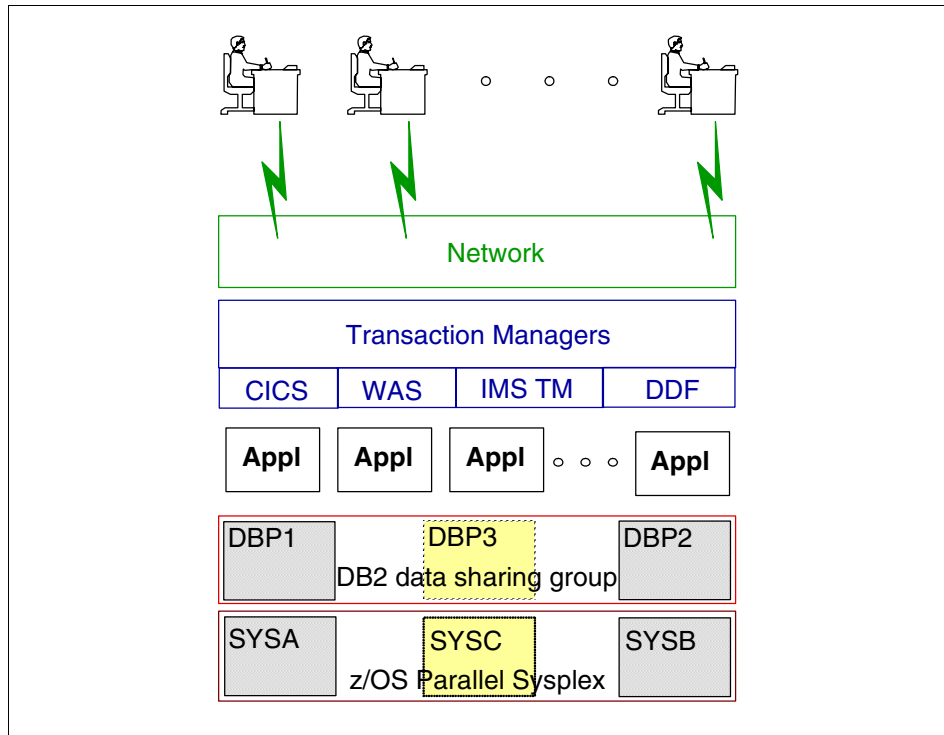


Figure 1-6 Capacity flexibility

### 1.3.3 Configuration flexibility

Data sharing lets you configure your system environment much more flexibly, making it easier to support the changing requirements for access to DB2 data. For example:

- ▶ You can have more than one data sharing group on the same Parallel Sysplex. You might, for example, want one group for testing and another group for production data.
- ▶ You can have more than one data sharing group in the same server, and even within the same LPAR.
- ▶ You can manage operational and decision support requirements separately even while maintaining all the data in a single data sharing group.
- ▶ You can optimize resources on one DB2 subsystem for one set of applications and favor a different set of applications on another DB2 subsystem. During your regular business day, you can favor query access on one DB2

subsystem, on-line transactions on a second subsystem, and batch processing on a third subsystem. Then, during the hours when query and on-line usage drop, you change buffer pool settings to favor batch on all three DB2 subsystems. You can do the same at z/OS image level.

- You can combine separate DB2 subsystems into a data sharing group to allow application access to both sets of data, thereby reducing the costs of managing the data and providing access to the data separately. One example where this could be helpful might be in the case of an acquisition. Another example might be where constraints in the past forced you to implement two production DB2 subsystems.

### 1.3.4 Higher transaction rates

Data sharing gives you opportunities to put more work through the system. As Figure 1-7 illustrates, you can run the same application on more than one member to achieve transaction rates that are higher than possible on a single DB2 subsystem. You can also run more than one application on the same member.

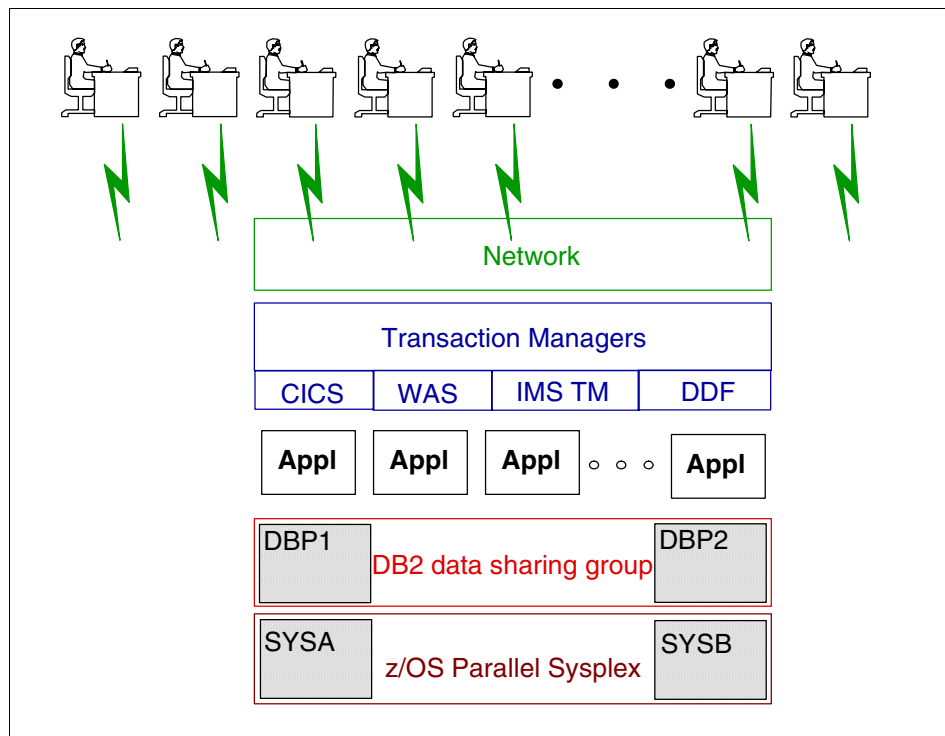


Figure 1-7 Increased transaction capacity

### 1.3.5 Application interface unchanged

Another benefit of data sharing is that the application interface is left unchanged. Your investment in people and skills is protected because existing SQL interfaces and attachments remain intact when sharing data. You can bind a package or plan on one member of a data sharing group and run that package or plan on any other member of the group.

## 1.4 Roadmap to implementing data sharing

You must complete a number of tasks to implement DB2 data sharing. Some tasks are done by your DB2 staff; other tasks are done by the z/OS, operations, or automations staff. In Table 1-1 we list these tasks, refer you to the sections of this redbook that discuss the tasks, and suggest which staff members might be participants in performing the tasks. We have placed these tasks in a logical order, but not all tasks must follow this order strictly. The interested parties will vary; your organization might have different interested parties for some or all of these tasks. Some of the tasks are beyond the scope of this book and in that case, the column titled “See section in this book” has been left blank.

These tasks apply to your production environment. You should adapt this list to suit your requirements for non-production environments such as system test or quality assurance. For example, one set of tasks in establishing your test or quality assurance data sharing group is identifying and creating test data, defining test cases, and determining success criteria for the tests.

*Table 1-1 Tasks to implement DB2 data sharing in your production environment*

No.	Task name	See section in this book	Interested parties
<b>A.</b>	<b>Planning tasks</b>		
1.	Establish naming conventions.	4.1, “Naming conventions” on page 66	DB2
2.	Determine DB2 configuration in the Parallel Sysplex.	3.4, “CF configuration alternatives” on page 56	DB2, z/OS
2.1	Ensure shared disk for DB2 system and user data sets.		DB2, disk
2.2	Review active and archive log considerations.	4.2, “Logging” on page 73	DB2
3.	Determine sizes of DB2 structures in the coupling facility (CF).	3.1, “Structure sizing” on page 40	DB2, z/OS

No.	Task name	See section in this book	Interested parties
4.	Plan for workload balancing.	5.1, "Objectives of workload balancing" on page 90	DB2, z/OS (WLM), automation, network, operations, scheduling
4.1	Identify connectors (DDF, CICS®, IMS, WebSphere®, and so on. Make sure each connector can take advantage of data sharing benefits.	Chapter 5, "Dynamic workload balancing" on page 89	DB2
4.2	Explore opportunities to make workloads portable.	2.5.1, "Portability" on page 36	DB2
4.3	Modify scheduling processes to reflect processing on more than one z/OS image.		Scheduling, operations
5.	Review application considerations.	2.5, "Application considerations in data sharing" on page 35	DB2, DBA, applications
6.	Plan for maintenance.	6.9, "Rolling maintenance" on page 119	DB2, z/OS, operations
7.	Modify automation tools to incorporate new DB2 members, new names and new processes.		DB2, automation
7.5	Update recovery procedures, and train operators in new procedures.		DB2/z/OS
8.	Determine TCP/IP port and resync ports.	4.1.6, "Distributed Data Facility (DDF) related parameters" on page 69	DB2, network
9.	Change specific DSNZPARMs.	4.3, "DSNZPARMs" on page 74 and 4.4, "Renaming an existing non-data sharing member" on page 78	DB2
10.	Plan for continuous availability.	In addition to those items listed below, see 2.5.1, "Portability" on page 36	DB2, DBA, z/OS, automation, network, operations, scheduling

No.	Task name	See section in this book	Interested parties
10.1	Restart-in-place procedures.	6.4, "Automatic restart manager (ARM)" on page 112	DB2, automation
10.2	Cross-system restart procedures.	6.4, "Automatic restart manager (ARM)" on page 112	DB2, z/OS, automation
10.3	Diagnostic procedures.	6.7, "Multi-system DB2 diagnostic dumps" on page 118	DB2, z/OS, operations
11.	Plan for monitoring the data sharing environment using current or new tools.		DB2, z/OS
<b>B.</b>	<b>Implementation tasks</b>		
1.	Change WLM policy definitions; include new DB2 member address spaces.	5.2, "Workload Manager" on page 92	z/OS (WLM)
2.	Define DB2 structures in the Coupling Facility Resource Management (CFRM) policy and start a new CFRM policy.	3.1, "Structure sizing" on page 40 and 3.5, "CFRM policy" on page 60	z/OS (DB2 input)
3.	Rename a DB2 subsystem.	4.4, "Renaming an existing non-data sharing member" on page 78	DB2
4.	Enable data sharing for the first DB2 member.	4.5, "Enabling the data sharing group" on page 85	DB2
5.	Add an additional data sharing member.	4.6, "Adding the second member" on page 87	DB2, z/OS, automations, network, operations, scheduling

No.	Task name	See section in this book	Interested parties
<b>C.</b>	<b>Post-implementation tasks</b>		
1.	Test data sharing function.		DB2, DBA
2.	Monitor behavior of a data sharing group.		DB2, z/OS
3.	Merge other DB2 subsystem into a data sharing group (optional).		DB2, DBA
4.	Run two DB2 members from the same data sharing group on the same z/OS image (optional).	7.7.1, "Multiple DB2 members in one z/OS image" on page 130	DB2, z/OS



## Data sharing architecture

DB2 data sharing is based on the Shared Data architecture, which differs significantly from other approaches that attempt to address capacity and availability challenges. The Shared Data architecture is fundamental to DB2's ability to meet those challenges and, at the same time, to provide unequalled scalability. The other benefits of DB2 data sharing — workload balancing, flexible configurations and systems management — are built upon the ability of the Shared Data architecture to meet the capacity, availability and scalability goals.

We discuss the following topics:

- ▶ Parallel database architecture alternatives
- ▶ Data sharing design for scalability
- ▶ Data sharing design for continuous availability
- ▶ Configuration flexibility and systems management
- ▶ Application considerations in data sharing

## 2.1 Parallel database architecture alternatives

Three general alternatives can be described as parallel database architectures. We call these alternatives Shared Nothing (SN), Shared Disks (SDi), and Shared Data (SDa). They are considered parallel because processes that access the data may execute in parallel. Figure 2-1 illustrates each of these alternatives.

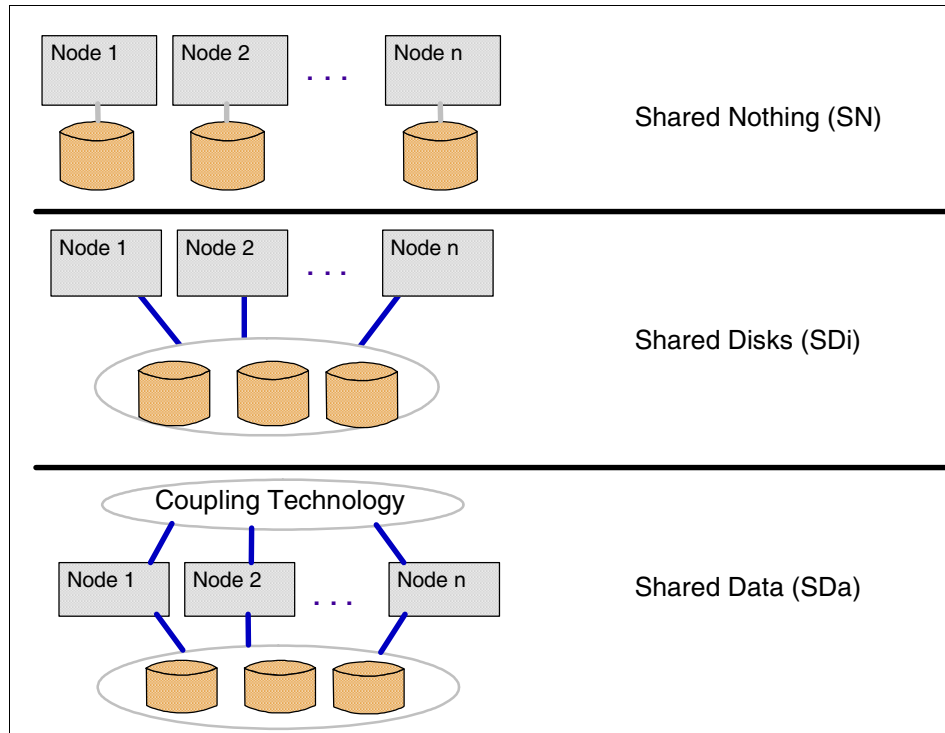


Figure 2-1 Parallel database architectures

### Shared Nothing (SN)

These are some characteristics of the Shared Nothing alternative:

- ▶ The database is partitioned.
- ▶ No disks are shared among the nodes.
- ▶ Distributed commit is necessary.
- ▶ Data re-partitioning is necessary as nodes are added.
- ▶ The partitioning scheme is susceptible to skewed access patterns.

The SN alternative allows accesses to be spread across nodes, which can provide a performance benefit for query workloads. To achieve this performance benefit, an application process has to be aware of the partitioning scheme and the location of the target data.



Availability is on a node basis; the loss of a node means loss of access to the data controlled by that node. Update processing that affects data in more than one partition requires a distributed commit. One node should take the role of coordinator, be informed when all the other nodes have committed, and report the result to the transaction manager. If nodes must be added to increase capacity, or if the changes to the amount of data are not uniform across nodes, then the data must be re-partitioned. Re-partitioning implies no application access to data, hence an interruption to application availability. The partitioning scheme cannot take into account multiple variations in access patterns, so performance can be skewed.

### **Shared Disks (SDi)**

These are some characteristics of the SDi alternative:

- ▶ Database partitioning is not necessary, but partitioning can give better performance.
- ▶ Fail-over characteristics of an SDi implementation are strong.
- ▶ Dynamic load balancing is achievable.
- ▶ Inter-node concurrency and coherency control mechanisms are needed; the messaging overhead limits scalability.

Each node can access all the disks, so the database does not need to be partitioned. In case of a node failure, application processes can access data from another node. The SDi alternative allows for dynamic load balancing if application processes can be shifted across nodes. Inter-node concurrency (locking) and coherency (buffering) controls are required to maintain data integrity. In the SDi alternative, these controls are based on messaging between the nodes. As nodes are added, this messaging can increase geometrically, which effectively limits the scalability of an SDi alternative.

### **Shared Data (SDa)**

These are some characteristics of the Shared Data architecture:

- ▶ SDa is an adaptation of SDi.
- ▶ Coupling facilities are used as hardware assist for efficient concurrency and coherency control.
- ▶ Continuous availability and load balancing capabilities are a design point of this configuration.
- ▶ Flexible growth is possible because additional nodes can be added without disruption to the applications.
- ▶ Messaging overhead is minimized, which yields excellent scalability.

The SDa architecture supported by the Parallel Sysplex is an adaptation of SDi, which uses coupling facilities (CFs) to solve the problem posed by messaging for concurrency and coherency control. High speed access to the CFs provides the necessary controls while minimizing messaging traffic. SDa is at least as robust as the SDi alternative for continuous availability and load balancing. Because of the coupling efficiencies and the ability to add capacity with low granularity, SDa offers greater flexibility for growth with excellent scalability.

DB2 data sharing exploits the SDa architecture to provide the database infrastructure for scalable growth, continuous availability, and flexible configuration.

## 2.2 Data sharing design for scalability

The two factors critical to preserving data integrity in a data sharing environment are inter-system *concurrency control*, or global locking, and inter-system *buffer coherency control*, or managing changed data.

Concurrency control is implemented in DB2 through global locking. *Global locking* is the process of allowing, within the data sharing group, multiple read operations *or* a single write operation to access a row or page of data.

Buffer coherency control, or managing changed data, addresses the situation in which one DB2 member changes data rows that already reside in the buffers of other members.

The cost of data sharing is attributable to the extra CPU cost needed to manage these two factors. Scalability and performance in data sharing depend on minimizing this cost.

The data sharing design points for scalability and performance include:

- ▶ Little or no performance impact if data are not actually shared, that is, if there is no inter-DB2 read/write interest.

*Inter-DB2 read/write interest* describes the situation where at least one member of a data sharing group is reading a table space or index and another member is writing to the table space or index.

If multiple members are only reading a table space or index, there is no need for concurrency and coherency controls. It is when at least one member writes to the table space or index that the controls are required.

- ▶ Dynamic recognition of sharing:

DB2, not an application, identifies when inter-DB2 read/write interest exists and when concurrency or buffer coherency controls are required. Once DB2

recognizes that inter-DB2 read/write interest exists, it initiates the necessary locking and buffer management processes. Applications do not have to be aware that inter-DB2 read/write interest exists.

- Minimal and acceptable CPU overhead if inter-DB2 R/W interest exists:

Establishing a data sharing group requires some overhead to establish global locking and buffer coherency controls. This overhead is minimized by using high speed access to structures in the CFs and by communicating primarily with the CF structures. Message traffic between DB2 members is used only to resolve contention situations.

- Near-linear scalability when adding the third through  $n$ -th nodes:

The primary overhead for managing global locking and changed data occurs when the second member is added to the data sharing group. There is very little additional overhead as additional members are added, generally less than one percent.

DB2 data sharing achieves this outstanding level of scalability based on its use of the lock structure and group buffer pools as described below.

## 2.2.1 Global locking and the lock structure

The internal resource lock manager (IRLM) is DB2's lock manager and assumes responsibility for global locking in a data sharing environment. Each IRLM in a data sharing group continues to manage local locks within the DB2 member. In addition, the IRLMs use the cross-system extended services (XES) component of z/OS to communicate with the CF and manage lock structure and global locking.

When IRLM needs to acquire a global lock on a resource, such as a table space partition or a page of a table, it uses the CF lock structure to acquire the lock. IRLM applies a hashing algorithm to the resource that needs to be locked. The result is the *hash class* for the resource. IRLM then issues a lock request to acquire the lock for that hash class. The lock structure in the CF determines if that hash class is available. If so, the lock request can be granted, and only this interaction takes place. With current technology, this interaction between IRLM and the CF using XES to communicate is a matter of microseconds ( $\mu$ sec).

When another DB2 member needs a lock on a resource, its IRLM issues a lock request for the corresponding hash class to the lock structure. As long as the lock requests can be granted, no messaging between IRLMs is required. Most lock requests can be granted without contention. Figure 2-2 shows lock requests from two nodes.

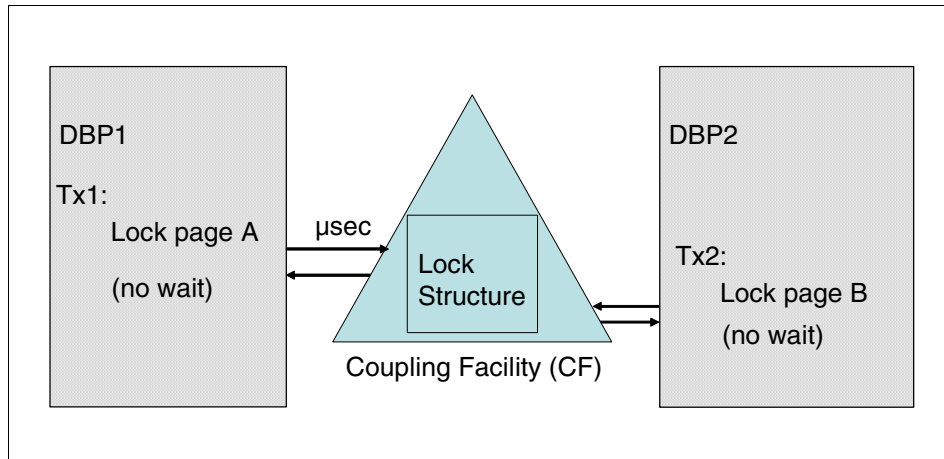


Figure 2-2 Locking in a data sharing environment

Only if there is a conflict, or *lock contention*, will one IRLM have to communicate with other IRLMs to resolve the conflict. In that case, the CF reports the contention to IRLM. IRLM will then exchange messages with *at most* one other IRLM to either resolve the contention or wait for the lock.

## L-locks and P-locks

DB2 and IRLM use two kinds of locks in a data sharing environment, logical locks and physical locks.:

- ▶ *Logical locks*, or L-locks, are locks held by transactions. These are the traditional locks DB2 has always used. In a data sharing environment, the IRLMs use L-locks to manage concurrency across the members of the data sharing group.
- ▶ *Physical locks*, or P-locks, apply only to data sharing. P-locks are part of the process to manage buffer coherency. For example, P-locks on data sets, such as table space partitions or index spaces, are used to determine when there is inter-DB2 read/write interest in the data set. Once there is inter-DB2 read/write interest in a data set, DB2 uses group buffer pools for the data set.

See 2.2.2, “Managing changed data and group buffer pools” on page 21 for further information.

## Locking optimizations

DB2 has the following optimizations, which reduce the necessity of processing locks beyond the local IRLM whenever possible:

- ▶ Explicit hierarchical locking avoids processing certain locks beyond the local IRLM when no inter-DB2 read/write interest exists in an object.

- ▶ If a single member with update interest and multiple members with read-only interest exist, IRLM propagates fewer locks than when all members have update interest in the same page set.
- ▶ All P-locks that are held beyond the local IRLM are owned by a member, not by an individual work unit. This reduces lock propagation by requiring that only the most restrictive lock mode for an object on a given member be propagated to the coupling facility. A new lock that is equal to, or less restrictive than, the lock currently being held is not propagated.
- ▶ IRLM can release many locks with just one request to the CF. This can occur, for example, after a transaction commits and has two or more locks that need to be unlocked. It also can occur at DB2 shutdown or abnormal termination when the member has two or more locks that need to be unlocked.

These optimizations reduce lock propagation and accesses to the CF, thus reducing overhead and enhancing the scalability of data sharing. For further discussion of how IRLM and CF manage global locking, refer to Chapter 6 of the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

## 2.2.2 Managing changed data and group buffer pools

The challenge of buffer coherency, or managing changed data, lies in the fact that DB2 buffers data. DB2 attempts to keep data or index pages in the buffers for potential reuse, reducing I/O and CPU costs. In a data sharing environment it is likely that two members of a data sharing group have many pages in common in their local buffer pools. If one member changes one of these pages, the other member has a back-level page. Buffer coherency consists of ensuring that the back-level page is brought current, to reflect the change, before the second member allows applications to access the page.

DB2 allocates cache structures in the CFs to manage buffer coherency. These structures are called group buffer pools (GBPs). When multiple members of a data sharing group open the same table space, index space, or partition, and at least one of them opens it for writing, the data is said to be of inter-DB2 read/write interest to the members. When there is inter-DB2 read/write interest in a particular table space, index, or partition, it is dependent on the group buffer pool, or GBP-dependent.

When a transaction changes a page of data, DB2 caches that page in the group buffer pool. The coupling facility invalidates any images of that page in the buffer pools associated with the other members. Then, if a request for that same page is subsequently made by another member, that member looks for the page in the group buffer pool.

Group buffer pools consist of *directory entries*, to track interest in a page, and *data elements*, to cache the changed page for high-speed access.

### Directory entries

Directory entries track pages of table spaces or index spaces that are GBP-dependent. When a member reads a page of a GBP-dependent data set into its local buffer pools, it registers that fact in the GBP by creating or updating a directory entry. One directory entry can track the registered interest in a page by all the members in the group. In addition, the directory entry tracks the specific location of the page in the members' local buffers and, if the page is written there, also in the GBP. This location information is used when a page is changed and the image of that page in the other members' buffers is invalidated. See "Cross-invalidation" on page 23 for a description of this process.

### Data elements

Data elements cache the changed pages of GBP-dependent data sets. This makes the pages available for high speed access from a member whose local image of the page is no longer valid.

DB2 ensures changed pages of a GBP-dependent data set are externalized to the GBP at commit. (It is also possible the page will be externalized before commit if local buffer thresholds are hit.) DB2 data sharing uses *store-in caching* to the CF, which means that DB2 assumes that a page written to the GBP is recoverable and the local buffer containing the changed page is stealable. One implication of store-in caching is that the changed pages occupying the data elements must eventually be written to disk. This process is called *castout* and is described below.

### Castout

Periodically, DB2 must write changed pages from the primary group buffer pool to disk. This process is called *castout*. Typically the responsibility of castout gets assigned in turn to every member of the group, with the castout work getting eventually spread evenly across all members. Each time, the member (called a *castout owner*) that is responsible for casting out the changed data uses its own address space because no direct connection exists from a coupling facility to disk. The data passes through a private buffer, not through the DB2 buffer pools.

See Figure 2-3 for an example with two members in the data sharing group. DBP1 and DBP2 will take turns in being assigned castout work.

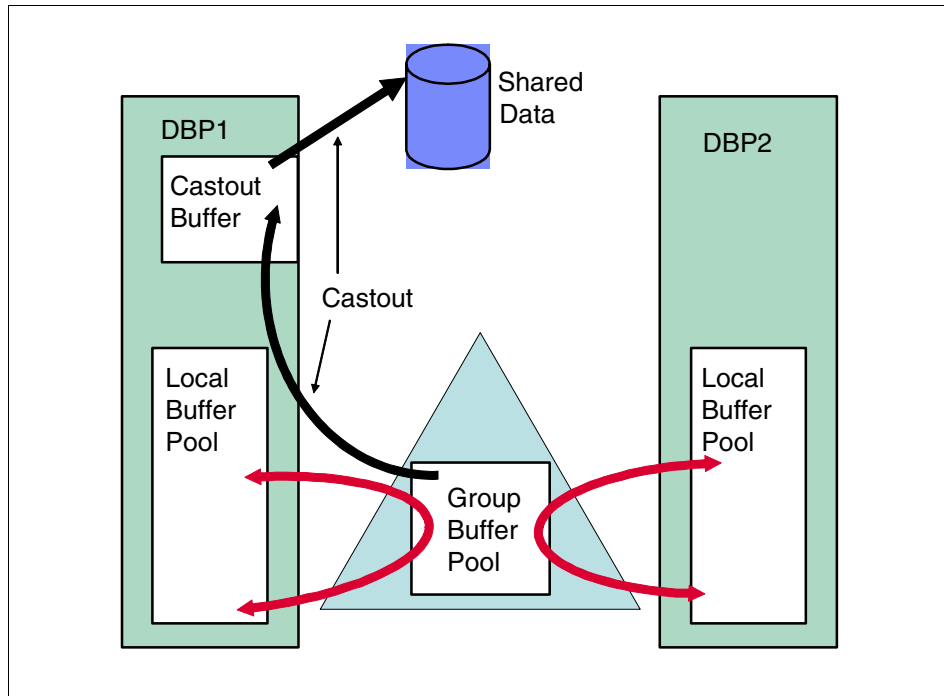


Figure 2-3 Castout process

Until castout has written a page to disk, it must continue to occupy its data element. Once a page has been castout, the data element in which it resides is a candidate for reuse.

## Cross-invalidation

When a changed page is externalized to the GBP, the directory entry indicates which other members have an image of that page and in what buffer locations those images are. The coupling facility control code (CFCC) sends a signal to those members that have an image of the changed page to invalidate that page in those members' buffers. This message is processed on the receiving systems without an interrupt.

When one of those members attempts to access the page, it detects that the image of the page in its buffer pool is invalid. The member then accesses the GBP to see if the page is still there. If it is, the page is read from the GBP to the member's local buffer pool. If the page is no longer in the GBP, the member reads it from disk.

## Managing changed data in a GBP: An example

Figure 2-4 illustrates the use of a group buffer pool for managing changed data. The numbers for each step below correspond to the numbers that appear in the diagram.

The scenario assumes that transaction Txn1 in DBP1 is already executing and has read page P1 into the local buffer pool in DBP1. Here is what transpires:

1. Transaction Txn2 begins executing and reads page P1 into the local buffer pool in DBP2. DBP2 registers P1 in a directory entry in the group buffer pool. This access occurs in a few  $\mu\text{sec}$ . We have lock avoidance.
2. Txn1 on DBP1 updates P1, completes the unit of recovery, and commits. During commit processing P1 is written to a data element in the GBP and the CF detects DBP2's interest in P1 and sends a cross-invalidate message. DBP2's buffer containing P1 is now invalid. DBP1 does not have to wait for a response from the cross-invalidate message.
3. Txn2 attempts to access P1. Because the buffer was marked invalid in step 2, DBP2 checks to see if P1 is still in the group buffer pool. If it is, DBP2 reads P1 from the GBP into its local buffer pool. If P1 is no longer in the GBP, DBP2 must read it from disk.
4. Asynchronously to Txn1 and Txn2, P1 is castout to disk through the current castout owner.

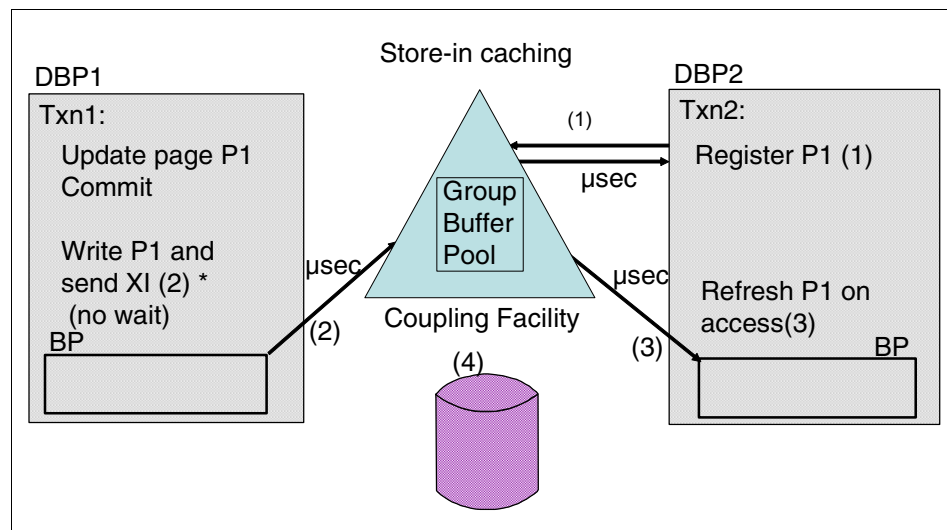


Figure 2-4 Store-in caching in the DB2 group buffer pools



## GBP optimizations

DB2 optimizes GBP access in several ways:

- ▶ Dynamic recognition of GBP-dependency means that page registration occurs only when there is inter-DB2 read/write interest. A data set is dynamically recognized as being GBP-dependent when the P-locks that IRLM manages indicate that at least one member is reading and at least one member is writing to the data set.
- ▶ If DBP1 has opened table space ABC for write (insert, update, or delete) and DBP2 has opened ABC for read, only DBP2 must register the pages of ABC that it reads into its local buffer pool. If DBP2 later opens ABC for write, then both DBP1 and DBP2 must register all pages of ABC that they read into their local buffers.
- ▶ If only one member has write interest, it is the only member to externalize pages to the GBP. In the scenario described in #2, above, only DBP1 writes changed pages to the GBP until DBP2 also opens ABC for write. Then both members will write changed pages to the GBP.
- ▶ Only those members that have an image of the changed page have pages marked invalid, and only those members that subsequently attempt to access an invalidated page need to refresh it.
- ▶ Castout occurs asynchronously to other GBP accesses. One benefit is that GBP accesses are not dependent on the speed of the disk devices.

These optimizations minimize GBP-related overhead and contribute significantly to the scalability of the data sharing architecture.

DB2 has additional performance options to fit your application requirements. By default, DB2 caches changed data, but you also have the options of caching all or none of your data.

You define group buffer pools by using coupling facility resource management (CFRM) policies. For more information about these policies, see 3.5, “CFRM policy” on page 60.

## 2.3 Data sharing design for continuous availability

The design goal for the data sharing architecture is to provide continuous availability across the planned or unplanned outage of any single hardware or software element.

The strategy to achieve this goal includes the capability to:

- ▶ Remove causes of planned outages.
- ▶ Build on the availability strengths of robust, fault tolerant System z™ and z/OS components.
- ▶ In a failure situation:
  - Isolate the failure to the lowest granularity possible.
  - Automate recovery.
  - Recover fast.

In the data sharing architecture, most single points of failure have been eliminated: DB2 subsystem or z/OS image, central processing complex (CPC), I/O path. In a Parallel Sysplex it is also important to eliminate single points of failure by having two Sysplex Timers (or a Backup Time Server with STP), two CF LPARs, and at least two CF links between each CF LPAR and each system in the Parallel Sysplex. The data sharing architecture provides for each of these resources to support failover to another corresponding resource. See 3.4, “CF configuration alternatives” on page 56.

Continuous availability for the components of the Parallel Sysplex, including hardware, software, and CF resources, provides the infrastructure for continuously available application access to data. The application software also has a role in achieving continuous availability. Refer to 2.5, “Application considerations in data sharing” on page 35 for further discussion of that topic.

## Planned outages

The data sharing architecture reduces an application’s planned outages by allowing access to data during planned change activity. Individual components may have a planned outage and still not require an outage by the application. That is the availability goal of the architecture.

The process of handling a planned outage is similar for a DB2 member, a z/OS image, or hardware. Changes to CF resources are slightly different. In these cases, we recommend that changes be scheduled for periods of low activity.

Figure 2-5 shows how an application can maintain access to data during a planned change for a DB2 member or a z/OS image. The first step is to direct any workload away from DBP1 or its z/OS image. Once all the work is running on other members of the data sharing group, DBP1 is ready for the change activity. When the change is complete, work is routed back to DBP1. The process is then repeated for DBP2 and any other members. This process is called *rolling maintenance* or *rolling IPLs*. This means the change affects one member at a time, the change is rolled through the data sharing group, while applications continue to execute.

Rolling maintenance can be applied to software maintenance or to release upgrades. The same approach can be used for changes in the hardware system environment.

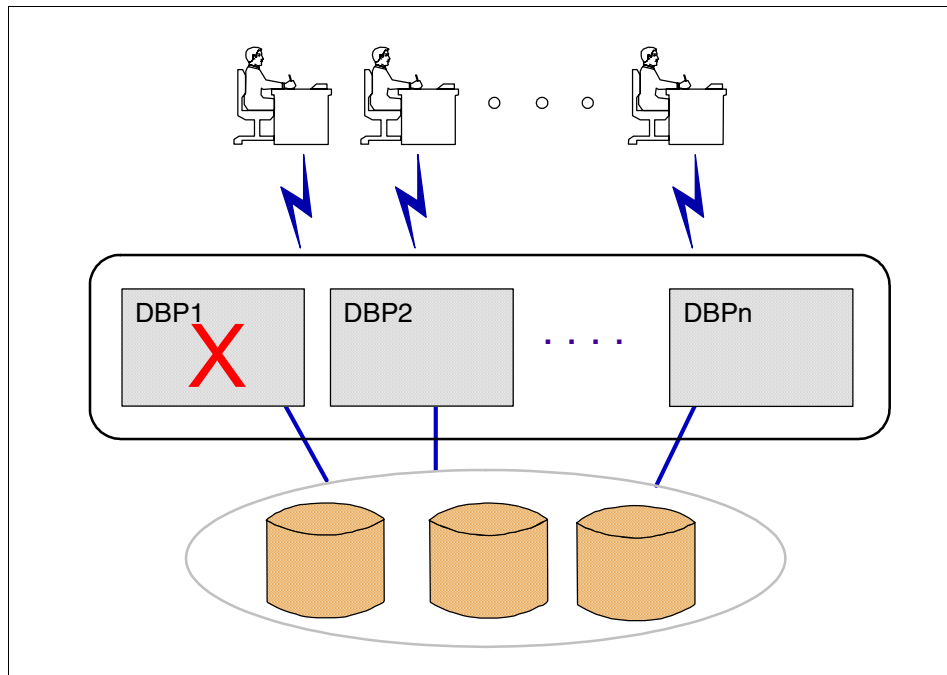


Figure 2-5 Planned DB2 or z/OS image outage

Planned changes to CF resources can include changes to the hardware system on which the CF runs, changes to the coupling facility control code (CFCC), changes to the amount of storage available to the CF, and changes to the CF links. Some of these changes can be made while the CF executes. Others require that the CF LPAR be deactivated. Many LIC patches can be made concurrently. The CF LPAR needs to be recycled for any changes to the CF driver level and to the total storage allocated to the LPAR.

Where the CF LPAR must be deactivated for the planned change, the structures on the CF can be “moved” to another CF LPAR. When a Parallel Sysplex is implemented, part of the capacity plan includes ensuring that there is sufficient space among the CFs for all the structures in any CF LPAR to be rebuilt to another CF LPAR. *Rebuild* is the term used to describe the process of moving a structure from one CF LPAR to another, and this can be done through a simple operator command, **REALLOCATE**.

Prior to deactivating the CF LPAR, the structures on the CF should be rebuilt to another CF LPAR. In DB2's case, this may include the lock structure, the *shared communications area* (SCA), and one or more group buffer pools. The SCA in the coupling facility contains information about all members' BSDSs and log data sets. Each of the DB2 structures should be rebuilt while the DB2 members and the applications continue to execute. Again, we recommend that these changes be scheduled for periods of low activity.

When the change is complete for the first CF and it is reactivated, all the structures can be rebuilt to the first CF, and the second CF can undergo the change. Finally, after activating the second CF, the structures can be rebuilt onto the second CF to restore the initial configuration using the POPULATECF command.

Figure 2-6 illustrates the situation where all the CF structures are allocated in one CF while the CF undergoes a planned change. Each of the DB2 members stays active and the applications have access to all the data.

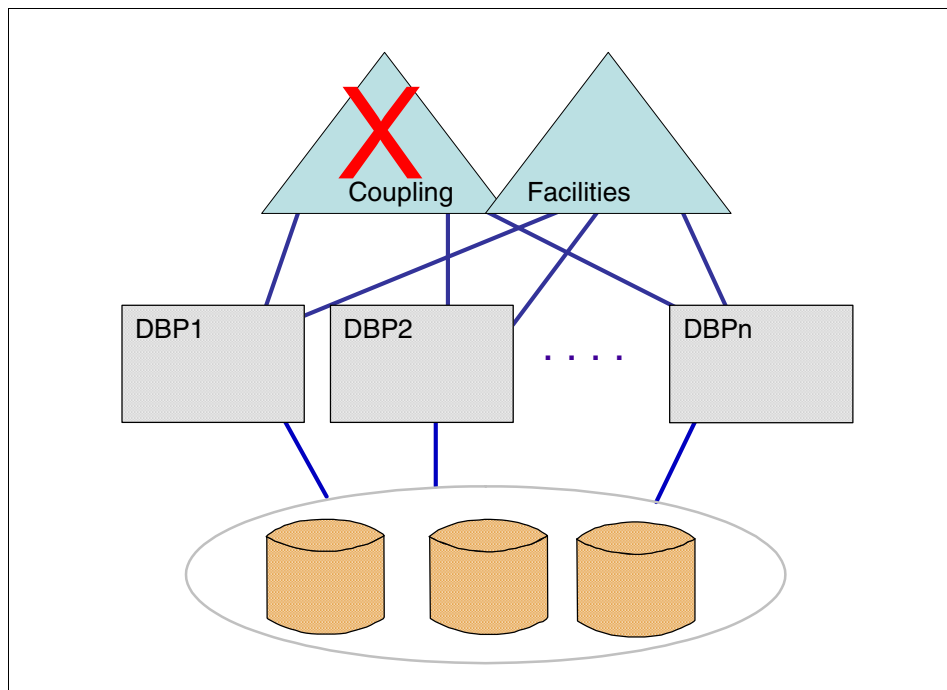


Figure 2-6 Planned CF outage

Another set of changes to CF resources includes changes to the structures within the CF. To accommodate growth or enhance performance, it may become necessary to change the size of a structure or change the CF on which the structure is allocated. In either of these cases, the DB2 structure can be rebuilt with the appropriate storage or in the appropriate CF LPAR while DB2 and the applications continue to execute.

See also 6.9, “Rolling maintenance” on page 119.

## Unplanned outages

Unplanned outages have the disadvantage of occurring unexpectedly. But that does not mean you cannot be prepared for them and avoid an application outage. The data sharing architecture is designed to allow new work to continue to be processed and for existing work to be recovered, and any lost resources to be restored, with minimal human interaction.

Figure 2-7 illustrates an unplanned outage to a DB2 member DBP1 or to the z/OS image where it executes. Whether DBP1 or z/OS experiences the outage dictates the recovery options, but each situation has similar impact to applications.

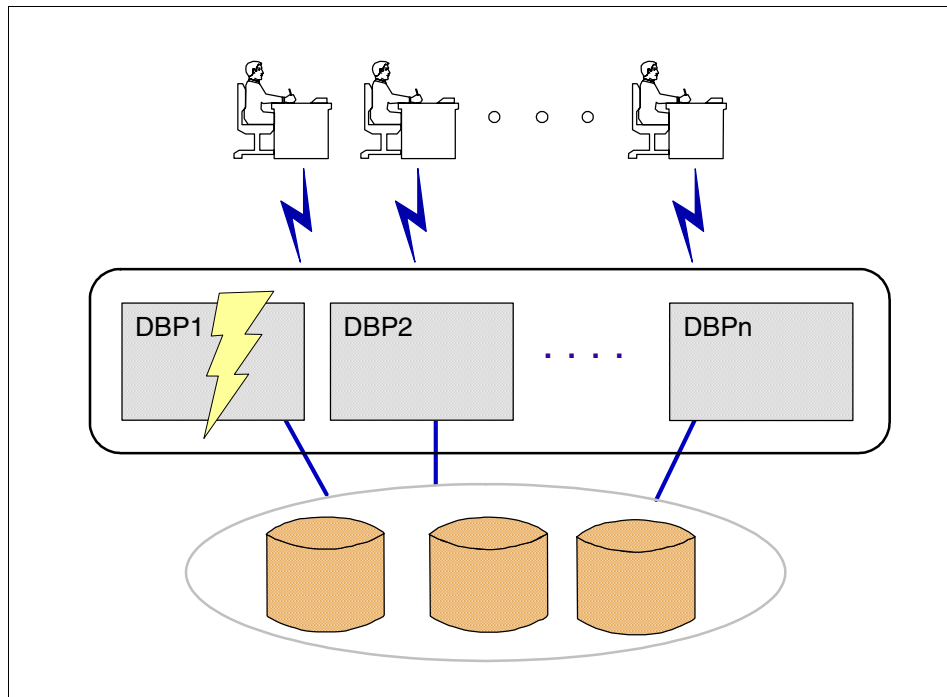


Figure 2-7 Unplanned outage to DB2 or z/OS image

As Figure 2-7 shows, application users continue to have access to data. New work entering the system will be routed to DBP2 or other members of the data sharing group. Transactions, tasks, or queries that were executing on DBP1 at the time of the failure are candidates for recovery.

DBP1 may hold locks on behalf of transactions that were executing at the time of the failure. If those locks were global locks, that is, if they were on resources that are actively shared, then the update type locks in the lock structure are changed to retained locks. This means that no other access is allowed to the resources protected by those locks until the underlying changes are either committed or backed out. The key to restoring application access to all resources protected by retained locks is to restart DBP1.

There are two basic scenarios: one that only affects DB2, the other that affects the entire LPAR or z/OS image.

- ▶ If only DBP1 or its IRLM failed, then restarting DBP1 on the same z/OS image will release the retained locks. The automatic restart manager (ARM) component of z/OS can initiate the restart quickest, but other automation options are available. During restart, DB2 will release retained locks. When restart is complete, work can be routed to DBP1, and your environment is returned to normal operation.
- ▶ If the z/OS image or the central processing complex (CPC) fails, DBP1 cannot restart in place. Because it is important to release retained locks quickly, waiting for an IPL of the z/OS image is not appropriate. Instead, DBP1 should be restarted on another LPAR that has connectivity to the coupling facilities. This is called a cross-system restart. ARM restarting DB2 with the DB2 *restart light* option is the fastest method, but other options are available. Restart light allows DB2 to release locks while taking minimal resources on the LPAR where it is started. When restart light completes, DB2 will shut down automatically. We discuss this further in 6.4, “Automatic restart manager (ARM)” on page 112, and 6.5, “Restart Light” on page 112.

Figure 2-8 illustrates an unplanned outage to a coupling facility LPAR. In a high availability configuration, the loss of a CF LPAR will not cause a DB2 or application outage. There may be brief spikes in response time or elapsed time during the CF recovery process, but the DB2 members, and the applications accessing DB2 data, will continue to execute.

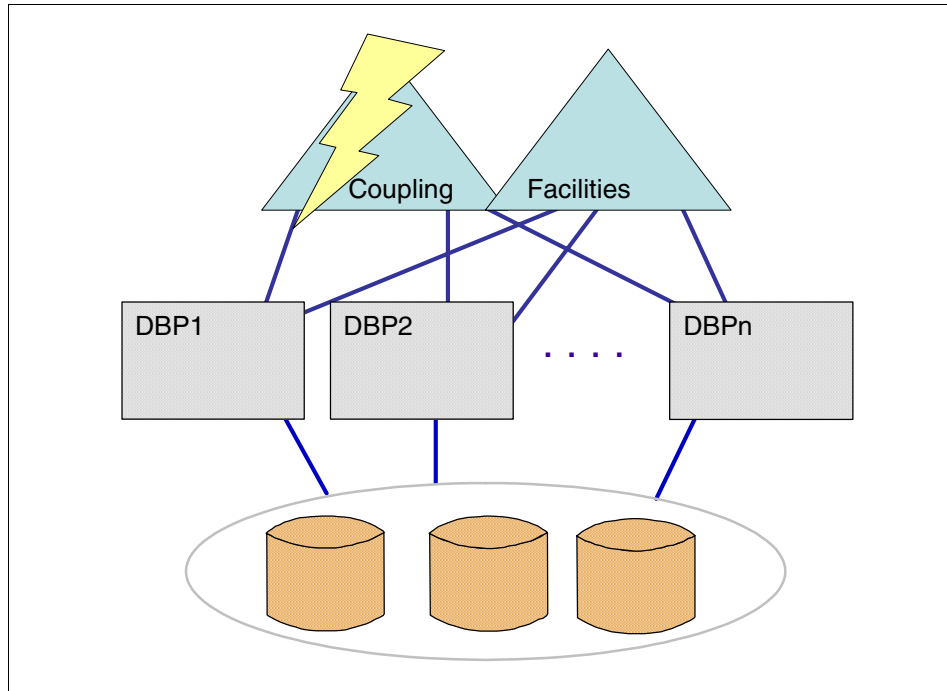


Figure 2-8 Unplanned CF outage

The DB2 and IRLM members cooperate to rebuild the structures to the other CF. See 6.2, “Component failure” on page 109 for more information on DB2 operations in failure situations

## 2.4 Configuration flexibility and systems management

Building on the scalability and availability features discussed above, the data sharing architecture provides significant configuration and systems management benefits. You can add members to your data sharing group, and systems to your Parallel Sysplex, in a wide variety of configurations. If you expect gradual business growth, you can add small increments of capacity and be confident in the ability of the new configuration to scale. Or you can add larger increments, for example, after a merger or acquisition.

You can move DB2 members and transaction managers around within the images in your Parallel Sysplex as long as your z/OS parameter libraries are shared. Figure 2-9 illustrates that the resources of the Parallel Sysplex can be modified without affecting the data sharing, application, transaction manager, or network layers.

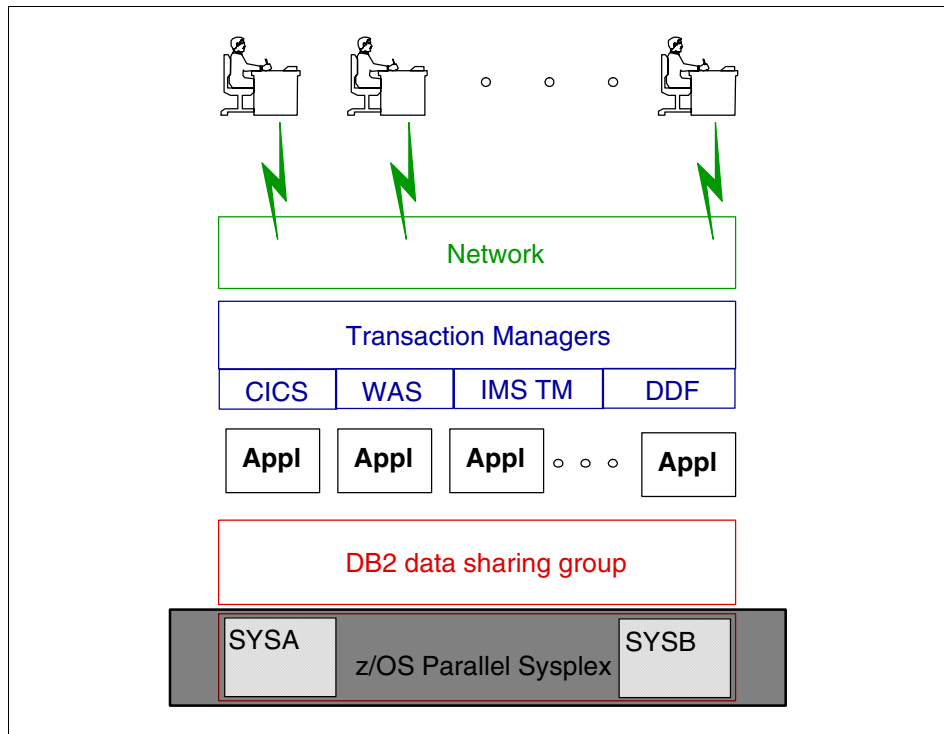


Figure 2-9 Flexible configurations for growth

Similarly, there is flexibility in the DB2 data sharing group layer, as illustrated in Figure 2-10. DB2 members can be started on any z/OS image that has access to the coupling facility and can be moved to take advantage of changes in the Parallel Sysplex infrastructure. Changes in the data sharing group layer do not necessitate changes in the application, transaction manager or network layers. Of course, if the DB2 changes offer performance or throughput advantages, it will be beneficial to make adjustments in the application, transaction manager, and network layers to exploit these advantages.



Workload balancing techniques allow incremental changes in the Parallel Sysplex or data sharing layers to be exploited by the other layers to optimize the resources in your environment (Figure 2-10).

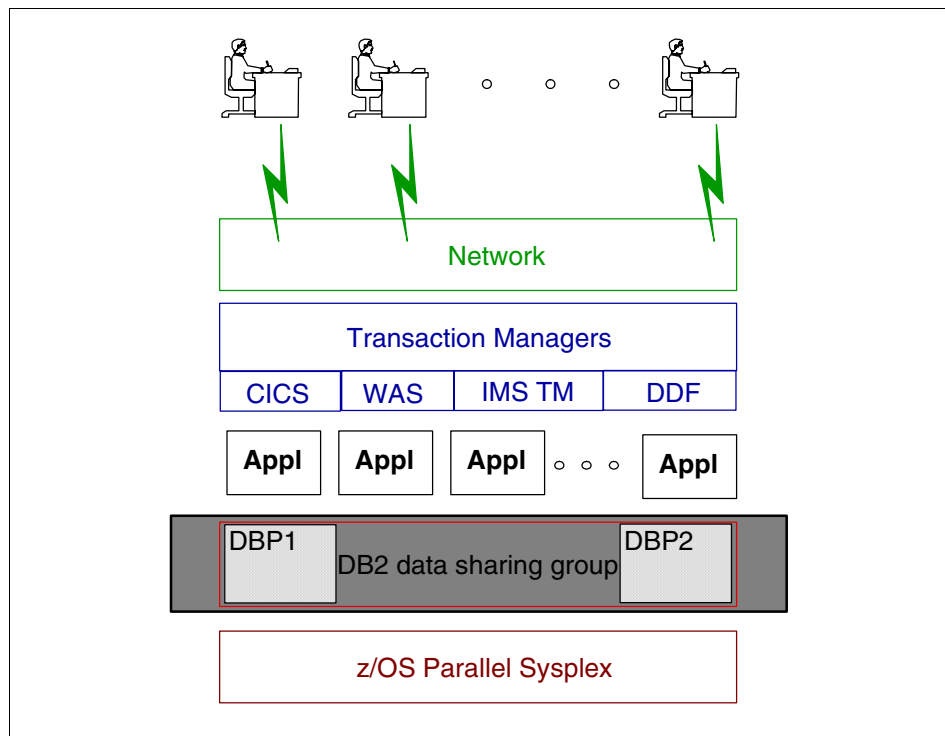


Figure 2-10 DB2 flexibility

Among the systems management advantages of data sharing is the ability to support operational and informational — or decision support — data and applications within the same data sharing group.

Often operational and decision support systems evolved separately, either due to capacity constraints or to concerns with managing the data to different service levels and with different access and security characteristics.

See Figure 2-11 for a diagram of data sharing implemented in this environment.

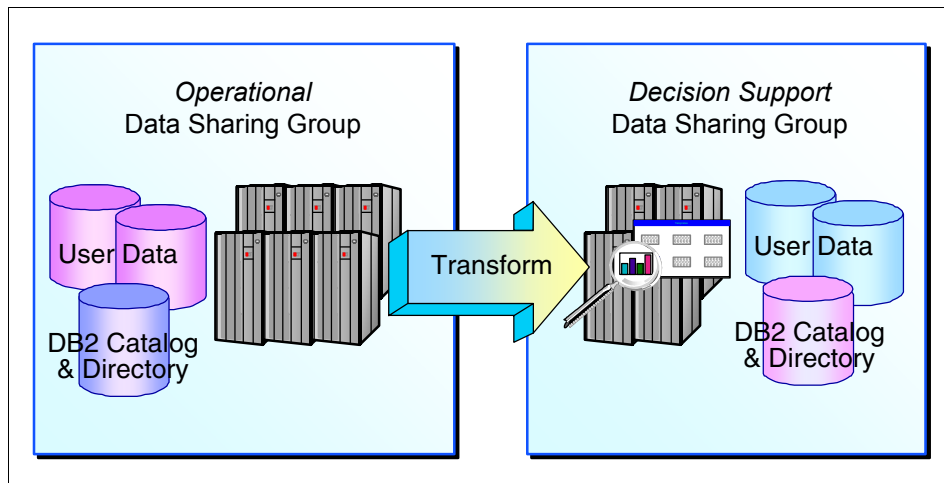


Figure 2-11 Separate operational and decision support groups

Data sharing allows you the option to bring the decision support data and applications back into the operational sphere. This allows greater flexibility for managing capacity within your environment, and enables the occasional requirements to join operational and decision support rows to be supported within a single production environment. You no longer have to maintain separate systems and rely on distributed database access when combining operational and decision support data. See Figure 2-12.

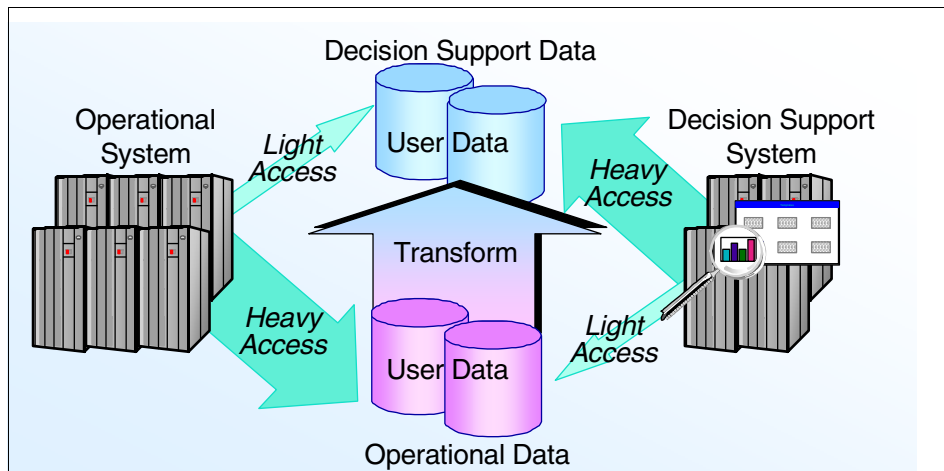


Figure 2-12 Combined operational and decision support data in one group

For further discussion of configuration flexibility and systems management, see Chapter 1 of the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

## 2.5 Application considerations in data sharing

One of the advantages of data sharing is that applications do not have to be aware that the DB2 environment has become a data sharing environment. In general, applications do not need to know upon which member they are executing, nor how many members are in the group, nor which other applications are currently executing. In addition, the SQL interface does not change, so most applications will execute in a data sharing environment without change.

Figure 2-13 illustrates that the application layer does not have to change to reflect changes in the data sharing group layer.

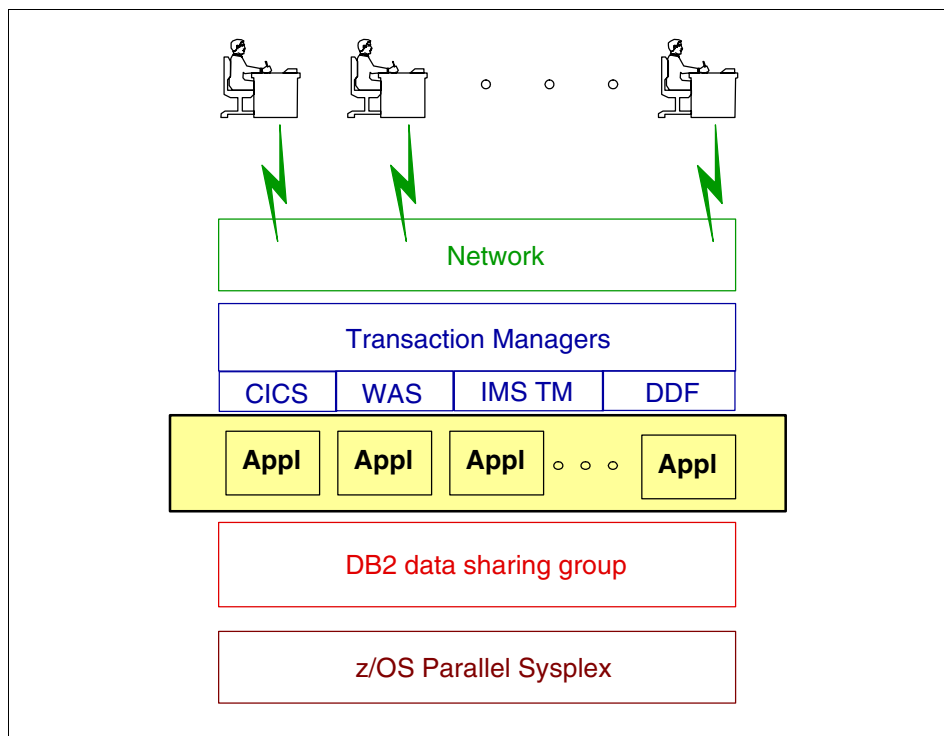


Figure 2-13 Applications run unchanged in data sharing

However, there are some facets of an application's behavior that may be candidates for change to take the greater advantage of data sharing's capabilities. These facets include portability, commit frequency and concurrency. Refer to *Parallel Sysplex Application Considerations*, SG24-6523, for a more complete discussion of these topics.

## 2.5.1 Portability

The continuous availability and dynamic workload balancing capabilities of data sharing will not provide much business benefit if application processes are not portable across members of a data sharing group.

*Application portability* in the context of data sharing means the ability of an application process (transaction, task, or query) to execute on more than one member of a data sharing group. If an application process is restricted to executing on a specific DB2 member, then that process is limited by the availability of that member and constrained by the capacity of that member.

To take advantage of the availability and workload balancing benefits that data sharing provides, an application should be able to execute on at least two members of a data sharing group. Thus, if one member is being migrated to a new release, the application process should be capable of running on the member of the group that is still active. If one member should suffer an unplanned outage, new occurrences of that application could be routed to the active member.

For optimal workload balancing, applications should be able to run on multiple data sharing members concurrently. This *cloning* of applications provides the greatest flexibility to handle workload spikes or changes in workload characteristics.

## 2.5.2 Commit and lock avoidance

Applications in a data sharing environment that meet best practices standards:

- ▶ Commit frequently
- ▶ Adjust their commit frequency based on external controls
- ▶ Include checkpoint/restart or retry logic
- ▶ Capture error messages
- ▶ Access common resources in a consistent pattern so as to avoid deadlocks

Most of these standards apply equally to non-data sharing environments. Commit frequency has additional implications in a data sharing environment. Long running update, insert, and delete processes lock resources that other processes may need to access. Also, inadequate commit frequency for long duration application processes — for example, batch jobs — may have the greatest impact on data sharing performance by reducing lock avoidance. Refer to 7.6, “The problem of runaway threads” on page 128.

*Lock avoidance* is the ability of DB2 to return rows to the application without holding a lock on the data row or page. Avoiding locks locally is good for performance; avoiding global locks can have a significant performance benefit. Refer to 7.5, “Lock avoidance: CLSN versus GCLSN” on page 128.

### 2.5.3 Concurrency

Concurrency of application processes means being able to execute more than one instance of the application at the same time. This is important in a data sharing environment because one process instance can execute on member DBP1, and another instance of the same process can execute on DBP2. Distributing these processes takes advantage of the scalability and workload balancing benefits of data sharing to increase the throughput of the entire Parallel Sysplex.

Applications ideally would include the possibility of cross-system execution in the design phase. For transaction or query workloads, cross-system concurrency is not very different from same-system concurrency: issuing frequent commits and accessing common resources in a consistent order. For batch type workloads, there are additional options and benefits, such as these:

- Reducing the overall batch window or the critical path of the batch job stream:

If your batch process runs in a single DB2 environment today, you may be able to dispatch some batch jobs to execute against the second member of your data sharing group. This will take advantage of the processor capacity of the second member. Jobs that execute in parallel in a single DB2 environment are natural candidates to dispatch on another DB2.

- Reducing the elapsed time of a specific batch process:

Some monolithic batch processes can be split into multiple jobs and executed against two or more DB2 members. For example, a batch job that updates a partitioned table can be split into multiple jobs, each job updating several partitions. These jobs could then execute on different DB2 members to reduce the overall elapsed time of the process. In this example, one consideration is that the input file may have to be sorted and split into multiple input files. Another consideration is that non-partitioned indexes on the table may experience global locking or buffer contention.





## The coupling facility

In this chapter we describe the basic data sharing implementation issues that concern the coupling facility. While the z/OS systems teams usually control most aspects of the CF, you (the DB2 team) must provide them with sizing information for the structures that DB2 uses.

All the information required by the z/OS or Parallel Sysplex team is given in this chapter.

We discuss the following topics:

- ▶ Structure sizing
- ▶ Auto Alter
- ▶ Duplexing
- ▶ CF configuration alternatives
- ▶ CFRM policy
- ▶ Best practices for the coupling facility

## 3.1 Structure sizing

DB2 structures are allocated to the data sharing group through definitions in the Coupling Facility Resource Management (CFRM) policy. Each parallel sysplex has one active CFRM policy. As we go through this exercise, we examine some examples of the STRUCTURE statement that your z/OS systems team will use in the definition.

The sizing strategy we employ is: “avoid shooting ourselves in the foot.” All joking aside, this phrase means that we would rather overcommit CF storage than to undercommit it when going into data sharing. After you have implemented data sharing, you can then monitor and tune based on your actual experience. We do not size for the minimum possible (even if we knew what this was). Rather, our structure sizes give you room to grow without a resizing effort as you gain experience and as your data sharing workload increases,

You can arrive at the sizes of the three DB2 structure types using the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 and other IBM z/OS manuals. The easiest way is through the use of a tool called CFSIZER. You can use it at this Web site:

<http://www.ibm.com/servers/eserver/zseries/cfsizer/>

Click **DB2** in the left hand column to get to the sizing information for data sharing.

First we will size the group buffer pools (GPBs) to illustrate the use of CFSizer.

### 3.1.1 Group buffer pools

We are assuming that you are sizing each GBP for a 2-way data sharing group. We also assume that you want both members to have the same local buffer pool (BP) sizes. From the buffer pool perspective, you can then shift workload from one member to another for planned or unplanned outages.



We are sizing the GBPs for *production*, assuming the use of the buffer pool configuration shown in Table 3-1 (test configurations do not have to be so exact).

Table 3-1 DB2 local buffer pools for group buffer pool sizing example

Buffer Pool	No. of Pages	Page Size	Notes
BP0	5,000	4 KB	DB2 Catalog and Directory
BP1	50,000	4 KB	Work data base
BP2	100,000	4 KB	
BP3	200,000	4 KB	
BP32K	10,000	32 KB	Required
BP16K0	5,000	16 KB	Required
BP8K0	5,000	8 KB	Required
And so on...			

When you click the following URL for CFSizer, you see the window in Figure 3-1:

<http://www.ibm.com/servers/eserver/zseries/cfsizer/>

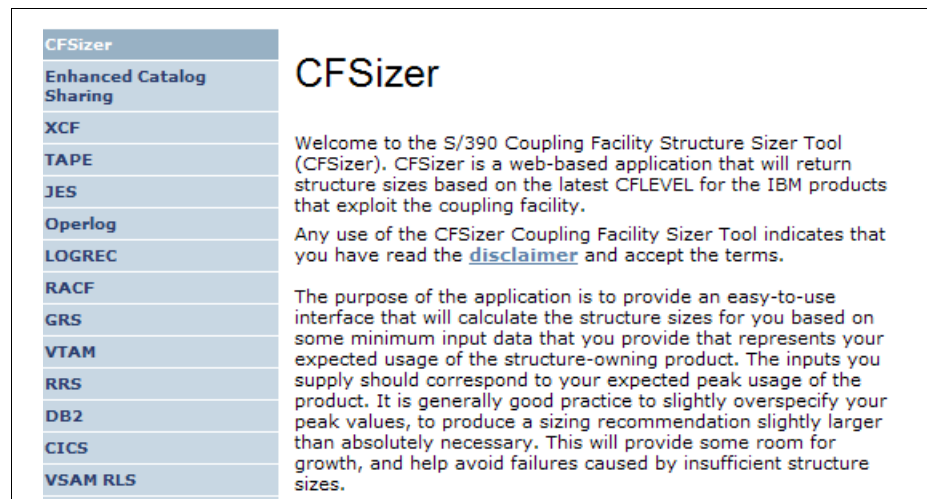


Figure 3-1 Partial CFSizer initial page

Click DB2 on the left hand panel and you see the window shown in Figure 3-2.

For form field help, click the Product/Help for that structure.

☐ DB2 IRLM lock structure [DB2 IRLM help](#)

Lock rate	Max number of connectors	# Table spaces
<input type="text" value="1000"/>	<input type="text" value="32"/>	<input type="text" value="1000"/>

☐ DB2 SCA list structure [DB2 SCA help](#)

# databases	# tables
<input type="text" value="50"/>	<input type="text" value="100"/>

☐ Group buffer pool 1 [GBP help](#)

Page size (4,8,16 or 32)	Number of buffers	GBP cache all/changed
<input type="text" value="4"/>	<input type="text" value="1000"/>	<input type="text" value="U"/>

☐ Group buffer pool 2

<input type="text" value="4"/>	<input type="text" value="1000"/>	<input type="text" value="U"/>
--------------------------------	-----------------------------------	--------------------------------

☐ Group buffer pool 3

<input type="text" value="4"/>	<input type="text" value="1000"/>	<input type="text" value="U"/>
--------------------------------	-----------------------------------	--------------------------------

[Submit](#)

Figure 3-2 Partial CFSizer screen for DB2

You can size three GBPs at a time. We shall now size GBP0, GBP2, GBP3. Remember that BP1, our work file data base buffer pool, is not shared, so it should not have a group buffer pool. Because we are sizing for a 2-way data sharing group, we multiply by 2 the buffer pool sizes shown in Table 3-1 on page 41.

Here is the procedure to follow:

1. Enter the following values in the **Number of Buffers** box:
  - 10000 under **Group buffer pool 1** (check its box)
  - 200000 under **Group buffer pool 2** (check its box)
  - 400000 under **Group buffer pool 3** (check its box)
2. Leave the **U** in each **GBP cache all/changed** box alone.

Refer to the panel shown in Figure 3-3.

☐ Group buffer pool 1
[GBP help](#)

Page size (4,8,16 or 32)	Number of buffers	GBP cache all/changed
<input type="text" value="4"/>	<input type="text" value="10000"/>	<input type="text" value="U"/>

☐ Group buffer pool 2

<input type="text" value="4"/>	<input type="text" value="200000"/>	<input type="text" value="U"/>
--------------------------------	-------------------------------------	--------------------------------

☐ Group buffer pool 3

<input type="text" value="4"/>	<input type="text" value="400000"/>	<input type="text" value="U"/>
--------------------------------	-------------------------------------	--------------------------------

Figure 3-3 GBP information entered in CFSizer

- Click **submit** (on the bottom of the screen) on Figure 3-3.  
The results are shown in Figure 3-4.

Servers > Mainframe servers > CFSizer >

## CFSizer structure size results

CFSizer structure size results (CF level 14)

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	15360	15360

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	292096	292096

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	582912	582912

Figure 3-4 CFSizer GBP allocations for three GPBs

These results are shown in **K** bytes. Notice that CFSizer does not show even numbers for INITSIZE, nor does it attempt to guess at SIZE. We come in at this point and do two things:

- We first round each number up to an even number (easier for us humans to remember)
  - GBP0 INITSIZE is 16,000
  - GBP2 INITSIZE is 300,000
  - GBP3 INITSIZE is 600,000

2. We then double that value for SIZE, to give us room for growth. The reason for that is explained in 3.2, “Auto Alter” on page 47.
  - GBP0 SIZE is 32,000
  - GBP2 SIZE is 600,000
  - GBP3 SIZE is 1,200,000

At the top of the Figure 3-4 are the words: **CFSizer structure size results (CF Level 14)**. The results shown were obtained by having CFSizer connect to an actual coupling facility to arrive at the structure size (not an estimate). The CF level referenced is the current level of the Coupling Facility Control Code (CFCC), which is microcode that allows a System z9 engine to be a coupling facility. Do not worry if your z/OS team tells you that the CF is not at that level. It will be eventually, and you want to be prepared for it.

**Important:** If there is not enough storage in the coupling facility to support these allocations, you can reduce the SIZE somewhat. Enough storage in the CF means that there is at least 10% free storage remaining if the SIZE values for all structures were reached. Your z/OS systems team can tell you if that is the case.

You can ignore the rest of the CFSizer window, which shows sample JCL that can be used by z/OS staff to implement the structures you have just sized.

3. Repeat the process using the remainder of the buffer pools in Table 3-1 on page 41. The input is shown in Figure 3-5.

Group buffer pool	Page size (4,8,16 or 32)	Number of buffers	GBP cache all/changed
<input checked="" type="checkbox"/> Group buffer pool 1	32	20000	U
<input checked="" type="checkbox"/> Group buffer pool 2	8	10000	U
<input checked="" type="checkbox"/> Group buffer pool 3	16	10000	U

[GBP help](#)

Figure 3-5 CFSizer input for the buffer pools that are not 4K page size

4. Click **submit** and you see the window shown in Figure 3-6.

## CFSizer structure size results

### CFSizer structure size results (CF level 14)

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	235008	235008

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	30208	30208

Function	Type	Structure name	INITSIZE	SIZE
DB2 GBP	CACHE	grpname_GBP	59648	59648

Figure 3-6 CFSizer results for buffer pools that are not 4 KB page size

Observe that the INITSIZE developed appears to be much larger for 10,000 pages for the 8 KB and 16 KB page GBPs than for the 4 KB page GBP of Figure 3-4 on page 43. The larger page size means that more space is allocated for the structure.

5. As we did with the previous GBPs, we round to even numbers for INITSIZE and double them for SIZE:
  - GBP32K is INITSIZE of 235000 and SIZE of 470000
  - GBP16K0 is INITSIZE of 60000 and SIZE of 120000
  - GBP8K0 is INITSIZE of 35000 and SIZE of 70000

While we have followed the rule of thumb of doubling INITSIZE for the SIZE values, you can use less for SIZE if these GBPs, which are required by DB2, are not used significantly by your DB2 subsystem.

6. We have accomplished the task of sizing the GBPs and you can now turn over the numbers you have developed to the z/OS Parallel Sysplex team for inclusion in the Coupling Facility Resource Management (CFRM) Policy. This team will actually allocate the structures. We show you an example of the statements they use in 3.5, “CFRM policy” on page 60. So far our STRUCTURE statements look like this:

```
STRUCTURE NAME=groupname_GBP0,INITSIZE=16000,SIZE=32000
STRUCTURE NAME=groupname_GBP2,INITSIZE=300000,SIZE=600000
STRUCTURE NAME=groupname_GBP3,INITSIZE=600000,SIZE=1200000
STRUCTURE NAME=groupname_GBP32K,INITSIZE=235000,SIZE=470000
STRUCTURE NAME=groupname_GBP16K0,INITSIZE=60000,SIZE=120000
STRUCTURE NAME=groupname_GBP8K0,INITSIZE=35000,SIZE=70000
```

Note that the size in a CFRM policy is in 1 KB blocks, whereas the size for BPs is in page size blocks.

### 3.1.2 Sizing the lock structure

The DB2 lock structure (defined as *group name\_LOCK1*) consists of two parts:

- ▶ The lock table, which is composed of lock table entries (LTEs), where global locks reside after going through an internal hashing algorithm
- ▶ The modify lock list, also known as record list entries (RLEs), where update type locks are kept

The size of the lock table is always an even power of two bytes wide.

The sizing of the lock structure could be accomplished with CFSizer, but there is an easier way, based on our experience with many DB2 users over the years. The size of the lock structure is small relative to the size of the other structures.

For 2-way data sharing, a safe value for INITSIZE is 64000. An exception to that recommendation exists for SAP and other ERP users. A common characteristic of these applications is heavy use of row level locking and very long commit scopes. You should follow the recommendations of the vendor for sizing in data sharing.

Our lock table is 32000K (half of 64000), and because it is divided by the 2-byte width, we have 16 million lock table entries (LTE).

If we specify a SIZE for growth, as we did for the GBPs, it only affects the half of the structure used for RLEs. The lock table can never be changed without rebuilding the structure. The RLEs can be increased through operator commands, up to the value of SIZE. This can come in handy for the rare instance that runaway jobs hold many locks and do not commit. Therefore you might want to increase the SIZE beyond 64000.

We show our lock structure statement information after adding a little extra for the RLE part of the table.

```
STRUCTURE NAME=groupname_LOCK1,INITSIZE=64000,SIZE=76000
```

When the first IRLM starts in data sharing, it divides the INITSIZE by 2 to arrive at the lock table size. It then rounds either down or up to the nearest power of 2, depending on which is closer. It requests the structure size from XES. The remainder of INITSIZE becomes the RLE or modify lock list part of the structure.

There is another way to tell IRLM exactly how many LTEs you want, and that is through the LTE keyword on the IRLM startup procedure. In addition to the STRUCTURE statement, on each of your two IRLM startup procedures, you can specify LTE=16 (million lock table entries). That specification tells IRLM to allocate that number of LTEs, rather than relying on the division process. Refer to the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, “Lock Structure Sizing” or *DB2 UDB for z/OS Version 8 Installation Guide*, GC18-7418 for more information.

Specification of LTE= gives you more control. It is particularly beneficial if, over time, your lock structure grows, but your RLEs do not. Continually doubling the lock structure size only for the purposes of increasing the LTEs wastes CF storage. (You normally would take this action if the false contention observed by the RMF™ coupling facility activity report shows a value greater than 1% of the total locks). Specification of LTE= allows a smaller lock structure size in this case.

The LTE specification takes precedence over STRUCTURE unless there is insufficient INITSIZE that it cannot be allocated and then should fall back to a 50:50 split.

### 3.1.3 Sizing the SCA structure

We prefer to size the SCA structure at 32 000K or 64 000K, to be generous. It contains exception status information, such as read only (RO) or LPL status for an object. Compared to the other DB2 structures, the SCA is infrequently accessed. It takes little storage in the CF. Here is our STRUCTURE statement:

```
STRUCTURE NAME=groupname_SCA,INITSIZE=64000,SIZE=64000
```

## 3.2 Auto Alter

The purpose of the GBP sizing we accomplished was to avoid two problems:

1. Cross Invalidation (XI) due to directory reclaim
2. Writes failed due to lack of storage

A third objective is to determine how many directory entries are needed for a given GBP. When DB2 is started for the first time, its defaulted ratio is 5:1, that is, five directory entries to every one data page. This value can be changed on a GBP basis by the user, but it is virtually impossible to “guess” at the ratio.

Figure 3-7 depicts two DB2s with BP1 and its corresponding GBP1. On the left is DB2A with local buffer pool BP1. It holds pages 1 and 2 from some table. On the right is DB2B with local BP1. It holds pages 3 and 4 of the same table. The GBP1, in the center of the figure, contains page 5 for the same table.

Figure 3-7 also has five very small blocks, named p1-p5. These represent directory entries for GBP1. There is an entry for every page. Observe that every page is different and that page 5 does not reside in either member's BP1. This is possible because when a DB2 member writes its page to the corresponding GBP, from the view of that DB2, the page is *as good as on disk*, and DB2 can use its local buffer to store any other page. Naturally, the page must be externalized to disk at some time through the castout process.

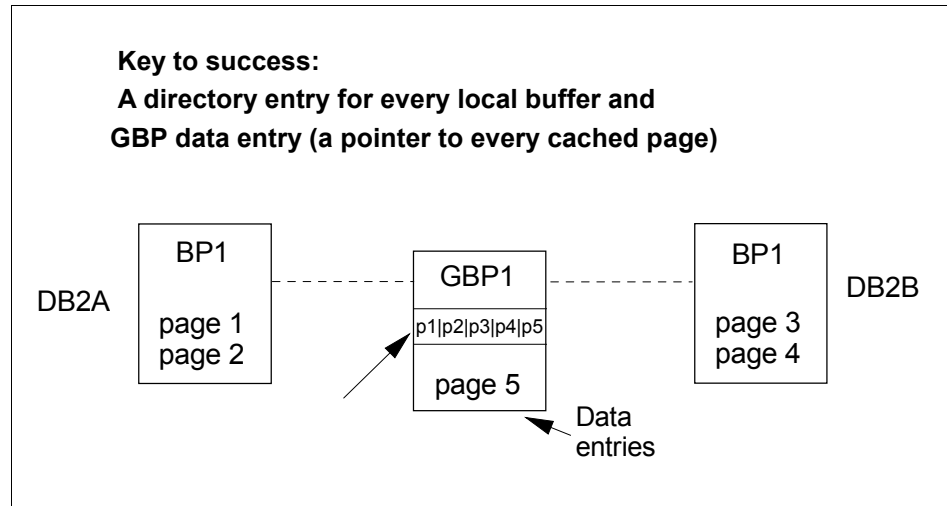


Figure 3-7 BP and GBP illustration

This represents the worst case situation, where there are no pages in the entire data sharing group that are the same. It is a conservative objective that we try to achieve.

The key to avoiding GBP directory entry reclaims is to have a directory entry for every different page that could be cached in a local buffer pool or in the associated group buffer pool. Generally speaking, this means having a directory entry for every "slot" (local buffer pool buffer or GBP data entry) into which a page can be read.

## Cross invalidations due to directory reclaims

Suppose that you have issued the command:

```
-DB2A DISPLAY GROUPBUFFERPOOL(GBP0) GDETAIL(*)
```



And DB2A reports:

```
....  
DSNB788I -DB2A CROSS INVALIDATIONS  
          DUE TO DIRECTORY RECLAIMS    = 4489  
          DUE TO WRITES                  = 3624  
          EXPLICIT                       = 0  
.....
```

Why are cross invalidations due to directory reclaims bad?

If a page is read from disk and the table is being shared (called GBP-dependent), it must be registered in the GBP. If there are so few entries that before the page is referenced, its entry is stolen for another page in the data sharing group, then the following processing occurs:

1. The page is referenced by DB2, but DB2 determines that the page is invalid.
2. DB2 then attempts to read the page from the GBP, but it is not there.
3. DB2 synchronously reads the page from disk and registers it (again) in the GBP

The overhead is two extra accesses to the CF structure and, what is worse, one synchronous I/O to disk. This activity is sometimes referred to as *thrashing* and is a sign of too few directory entries and likely a too small GBP.

## Writes failed due to lack of storage

From an availability standpoint, there is another serious problem that you should avoid.

The DB2 **-DISPLAY GBPOOL MDETAIL** command reports some *failed writes*, as shown in the following coding example:

```
DSNB777I - ASYNCHRONOUS WRITES  
          CHANGED PAGES                  = x  
          CLEAN PAGES                   = y  
          FAILED WRITES DUE TO LACK OF STORAGE = z
```

This means that there were not enough pages available in the GBP to accept a write from the local buffer pool. The write is attempted a few times while DB2 goes through the castout process in an attempt to free more pages. However, after a few unsuccessful attempts, DB2 gives up and inserts an entry for the page in the logical page list (LPL) where it is unavailable for any access until the LPL condition is removed. DB2 can automatically initiate the LPL recovery or the pages can be removed from the LPL manually by issuing the **-STA DATABASE** command.

### 3.2.1 Implementing Auto Alter

We can avoid both situations and also ensure that there is a proper directory to data ratio for each GBP by use of a z/OS capability called *Auto Alter*. This capability is initiated by the z/OS structure full monitoring that occurs regularly for each structure. Structure full monitoring adds support for the monitoring of objects within a coupling facility structure. This type of monitoring determines the level of usage for objects within a coupling facility, and issues a warning message if a structure full condition is imminent. Doing this allows tuning actions to avoid a structure full condition. Structure full monitoring occurs every few seconds.

Auto Alter has algorithms that can request the CF to dynamically increase or decrease the number of entries and/or data page elements to avoid structure full conditions. It can increase or decrease the size of the structure if necessary.

**Note:** Auto Alter's design point is for gradual growth, not to handle spikes in the workload.

Auto Alter applies to all CF structures. For the lock structure, it can only dynamically increase the RLE portion of the structure. It can increase the size of the SCA. Its main value in DB2, however, is for the GBPs. Guessing at a proper directory to data ratio is almost impossible. A GBP usually needs more than a 5:1 ratio. Static values might or might not be accurate and, over time, they could change as the workload changes. Auto Alter can pinpoint precisely the ratio needed at any given time and change the ratios dynamically. By comparison, if the ratio is changed by operator command, the structure must be manually rebuilt, which causes a slight disruption.

When either the directory entries or data pages exceed the FULLTHRESHOLD, XES will increase the size of the structure and will increase the component in short supply while decreasing the other one. For DB2 GBPs, the shortages usually occur for the data elements, not directory entries.

Note that if there is general storage stress on the CF (less than 10% free storage), XES can decrease those structures with ALLOWAUTOALTER(YES). XES will never decrease them below MINSIZE (defaulted to 75% of INITSIZE).

Even though XES is directing the CF to make the changes, the DB2 **-DISPLAY GBPPOOL** command reflects the current directory to data ratios. The total numbers of directory entries and data elements are found in the CF usage summary section of the RMF CF activity report.

For more information, you can refer to the z/OS manual, *Setting up a Sysplex*, SA22-7625, section titled "Allowing a Structure to Be Altered Automatically."

## Auto Alter capabilities

These are the main Auto Alter capabilities:

- ▶ Auto Alter supports autonomic tuning, because you set the sizes and Auto Alter does the rest.
- ▶ Auto Alter alters ratios in the CF without rebuilding the structure (versus the DB2 **-ALTER GBPPOOL** command that requires a rebuild).
- ▶ Auto Alter builds a better directory to data ratio than manual tuning (up to 40:1). Reclaim avoidance adjusts dynamically to changes with workload.
- ▶ Auto Alter can avoid the error: Writes failed due to lack of storage.
- ▶ Auto Alter allows DBAs to change local BP sizes or add a member without needing a CFRM policy change. It adjusts to gradual growth in the workload.

The way we have sized our GBPs is for proactive tuning, where there is enough storage in each structure to allow for growth, either through increases in local BPs or by adding another DB2 member. The goal is to allow GBP tuning to be ignored for a certain amount of time, perhaps six months to a year. This approach is safe when there is sufficient storage available in the CFs.

**Important:** Your z/OS systems teams must allow enough *white space* in each CF to allocate all of the structures in both CFs. In the rare event that a CF fails, for whatever reason, the structures should rebuild quickly in the remaining CF. This is specified in the STRUCTURE statement via the REBUILDPERCENT value of 1, and the PREF keyword, where at least one other CF is referenced.

Once we have sized our structures, it is easy to start using Auto Alter. We simply enter the parameter ALLOWAUTOALT(YES) with each STRUCTURE statement for which we want to allow structure alteration. Example 3-1 shows the GBP0 definition.

### *Example 3-1 GBP0 definition*

---

```
STRUCTURE NAME=groupname_GBP0
INITSIZE=16000
SIZE=32000
ALLOWAUTOALT(YES)
FULLTHRESHOLD=80
MINSIZE=16000
PREFLIST=(CF2,CF1)
DUPLEX(ENABLED)
REBUILDPERCENT(1)
```

---

We do not want the GBP to fall below the INITSIZE we have specified, so we code MINSIZE with the same value as INITSIZE. We also set the FULLTHRESHOLD to 80% (also the default).

## Monitoring the GBPs

We wish to monitor for the two values “Cross Invalidations due to directory reclaims” and “Writes failed due to lack of storage”. The easiest way to do this is through the DB2 command:

**DISPLAY GBPOOL(\*) GDETAIL(\*) TYPE(GCONN)**

This command can be issued from any *one* member that has connectivity to all the GBPs. **TYPE(GCONN)** lists only those GBPs that are allocated, reducing the output greatly from a list of all those possible, most of which are unallocated. **GDETAIL(\*)** provides the statistics from the time the GBPs have been allocated, which could be days, weeks, or months. We expect to see zeros for each of the two thresholds.

If you issue the command with **GDETAIL** parameter without the asterisk, then you get the statistics from the last time you entered the command *on the same member*. You can use this form of the command if you have your automation issue this periodically (every shift, every day) to determine the incremental changes.

As the GBP size is increased by Auto Alter, your z/OS systems staff should be aware of the increase and change the INITSIZE STRUCTURE statements in the CFRM policy. Otherwise, when a new policy is activated, the current sizes revert to INITSIZE and presumably would be increased quickly by Auto Alter.

**Note:** The current directory to data ratio is kept in the BSDS and does not revert to the 5:1 ratio.

## GBP thresholds

The following DB2 V8 defaults are recommended as GBP best practices for the current processor generation:

- ▶ GBP checkpoint interval: 4 minutes - Use an interval of 4 minutes.
- ▶ GBPOOLT: 10-30% - Use the larger number for GBPs < 500,000-1,000,000K.
- ▶ CLASST: 2-5% Use the larger number for GBPs < 500,000-1,000,000K.

Do not use 1% unless your **-DIS GBPOOL** command output shows at least 12,800 *pages*, *not INITSIZE*. DB2 tries to cast out 128 pages at a time.

**Note:** Do not use 0% (in case you are matching your VDWQT for local buffer pool thresholds). While the 0% value is designed to cast out 40 pages, it will choose as a castout threshold either 40 pages or 1% of the number of pages in the GBP, whichever value is *smaller*. If your **-DIS GBP00L** command output shows that you have 450 *actual pages*, you would trigger castout at 5 pages, which would be far too often.

The idea behind these recommendations is to trickle castout at a reasonable cost, so the other thresholds trigger castout instead of the GBP checkpoint. For large GBPs, the checkpoint would castout many pages at once.

If you knew exactly which type of data is in each GBP (such as frequently re-referenced, never re-referenced) you can choose better thresholds than those we describe. Our experience is that most users do not know or the content is mixed (BP2 for indexes) or organized functionally (BP3 for partitioned table spaces, BP4 for Application A), so the general case will be best.

In order to minimize the occurrence of GBP checkpoints at the same time, you could use prime numbers as time intervals, such as 3, 5, 7 minutes, depending on size and activity.

### 3.3 Duplexing

Structure duplexing was implemented so there is a second copy of the lock and SCA structures available for immediate access by all members of the data sharing group. It is strictly an availability option.

There are two forms from the view of z/OS XES component:

- ▶ User-managed (DB2) duplexing:  
User-managed duplexing is implemented only by DB2. From the duplexing view, DB2 is the “user” and it is able to optimize duplexing to minimize overhead, while providing for structure integrity.
- ▶ System-managed duplexing:  
System-managed duplexing is implemented by XES and is applicable to other (non-DB2) structures in the coupling facilities. System-managed duplexing is not used for the DB2 GBPs, but it applies to lock and SCA structures.

The two copies of the structure are referred to by different names in publications. DB2 and RMF reference them as *primary* and *secondary*, while z/OS publications and displays call them *old* and *new* respectively.

### 3.3.1 Group buffer pools (GBP) or user-managed pools

Group buffer pool duplexing was introduced in DB2 V5, in the late 1990s. It was important because a total CF failure involved lengthy recovery and there were no other alternatives at the time.

Without duplexing, there are two conditions that can occur if there is a disruption to the CF or its connections:

- ▶ Condition 1: If there is connectivity lost between a CF and one of the members of the data sharing group, but there is at least one other member that can read the GBP, the condition is detected automatically by a DB2 with connectivity. If the STRUCTURE REBUILDPERCENT parameter is coded (we recommend 1%) and it is less than the SFM weight, then a DB2 with connectivity *rebuilds* the structure in the other CF named in the preference list (STRUCTURE PREF keyword). This operation occurs quickly, normally within a couple of minutes, and with minimal disruption. Situations causing this condition occur when there is link failure or deactivation. This is one reason why we strongly recommend two CF links per image to each CF.
- ▶ Condition 2: The situation is different if the CF is completely inaccessible to any of the members. No member can read the structure and initiate the rebuild. If a CF fails, *all data sets that have pages in any GBP will be placed in group buffer pool recovery pending (GRECP)* and are unavailable until they are recovered. While DB2 will automatically recover them (if this situation does not occur during DB2 restart), it can take from seconds to minutes (up to an hour), depending on how many data sets are set to GRECP. The DB2 logs since the last GBP checkpoint (should be less than 4 minutes of log) are read and updates are applied from them.

If GBPs are duplexed and a failure occurs, fallback to simplex occurs within seconds, non-disruptively. There will be no rebuild activity if a link failure alone occurs. There should be no GRECP or LPL activity that occurs unless both CFs cannot be accessed.

While there are two copies of the GBP, DB2 achieves efficiency by limiting activity to the secondary GBP to the writing of changed pages, which is performed asynchronously. The activities that DB2 performs only for the primary GBP are: page registration, reads into local buffer pools, and castouts. For this reason, the overhead of writing to the secondary GBP might be only 10-50% the cost of the primary. You might see the higher number when there is heavy batch updating occurring. The lower one is for an OLTP workload. For most customers, 90% of the accesses to the CF are for page registration (an activity that does not occur for the secondary GBP), because many more pages are read than are updated.

**Important:** The additional cost quoted for writing the secondary GBP is *10-50% of the cost of the initial writes to the GBPs*. For example, if the total system effect of GBP CF activity is, for example, 5%, then the added cost for duplexing the GBPs is computed as follows:

- ▶ For light updating OLTP workloads:
  - Multiply 10% times the 5%, yielding .5% additional CPU for duplexing
- ▶ For batch updaters:
  - Multiply 50% times the 5%, yielding 2.5% additional CPU for duplexing

Confusion sometimes occurs because the cost of duplexing the GBPs is *significantly less* than the cost of system-managed duplexing a structure like the lock structure.

**Best practice:** It is a strong and *best practice* recommendation that all GBPs be duplexed.

If a failure and lengthy recovery occurs without duplexing, a post mortem review where the situation could have been prevented is hard to explain.

### 3.3.2 System-managed duplexing

System-managed duplexing was introduced by OS/390 V2R8 and implemented by DB2 for the SCA and lock structures in DB2 V7 in 2001.

System-managed duplexing provides a second copy of the structure that is an exact copy of the primary structure. When a CF fails, fallback occurs to simplex within seconds, avoiding an outage.

The following requirements, which are external to the normal concerns of the DB2 staff, are necessary for duplexing:

- ▶ Provide CF to CF links that allow communication and synchronization to occur between the CFs involved.
- ▶ Ensure that there is sufficient host capacity (in the z/OS processors) for duplexing overhead.
- ▶ If your CF CPU utilization as reported by RMF CF activity report is 25% peak, we recommend adding a second engine. Refer to 7.4, “How many CF engines are necessary?” on page 126.

Roughly speaking, a duplexed lock can easily take 3 times the cost of a simplexed DB2 lock request. For example, if a synchronous lock takes 10  $\mu$ sec, then a duplexed lock, when performed synchronously, would take about 30  $\mu$ sec. It is likely that the lock would be converted to asynchronous in order to spare the host processor, which must spin while asynchronous activity is performed. The asynchronous lock is likely to take more than 100  $\mu$ sec at the least.

For more information, refer to *System-Managed Coupling Facility Structure Duplexing*, GM13-0103, at:

<http://www.ibm.com/servers/eserver/zseries/library/techpapers/gm130103.html>

You can find another related white paper, *System-Managed CF Structure Duplexing Implementation Summary*, at:

<http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130540.pdf>

## 3.4 CF configuration alternatives

Figure 3-8 shows the possible CF configurations.

They are identified as follows:

- ▶ The two external (to the DB2 members) CFs are shown in “A”.
- ▶ One external CF and one internal CF are found in “B”.
- ▶ Two internal CFs on processors are represented by “C”. There are DB2 members whose structures reside on those CFs.
- ▶ Two internal CFs are shown in “D”, but there is no DB2 member on the processor with ICF02.



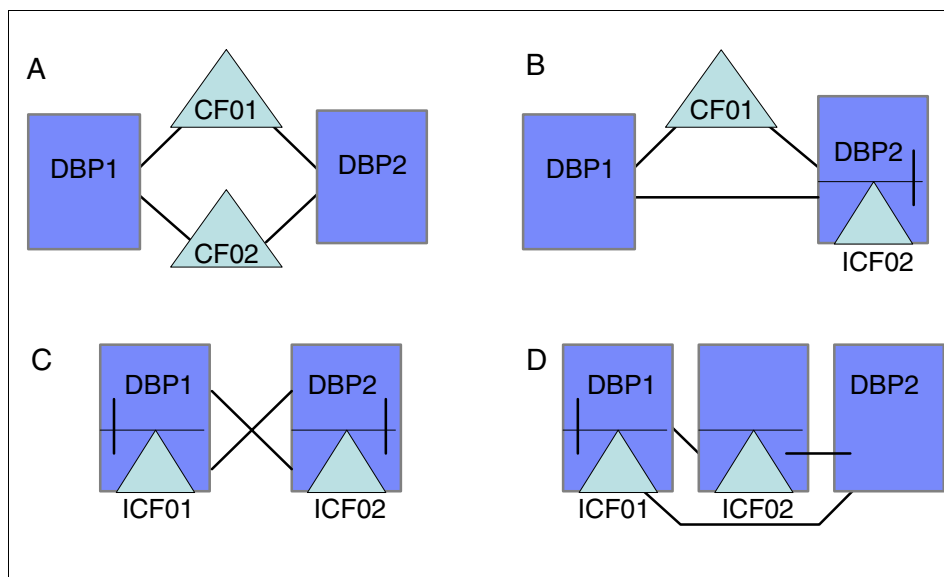


Figure 3-8 Four possible CF configurations

The System z processors are distinguished by many fast processor engines in the box and a number of speciality engines. The specialty engine important in this section is the internal coupling facility (ICF). ICFs have a number of financial advantages. They are not counted as a general purpose processors, because they can only run the coupling facility control code (CFCC), the microcode that turns them into CFs. ICFs require fewer external links than external CFs, because they are internal to the z/OS images on the same box. Configuration “C” in Figure 3-8 is the most common configuration for these reasons.

Parallel Sysplex has fast recovery from single failure. If a single CF fails, the DB2 lock/SCA structures should be rebuilt into the other CF. Any GBPs in the affected CF fall back to simplex.

### 3.4.1 ICF-only: Double failure for the lock and SCA structures

We assume that you have a 2-ICF configuration, where there are no external CFs, represented by “C” in Figure 3-8. We assume that the lock and SCA structures are placed in the same CF due to similar failure characteristics.

Fast rebuild usually implies that each affected member has its own part of the structure contained in its own virtual storage. Thus rebuild can occur quickly, within a couple of minutes upon failure of a single CF.

If a double failure occurs, recovery is not so quick. A double failure refers to a CF failure and the failure of at least one z/OS image containing a DB2 whose structures reside in the failed CF.

Assume that the entire processor containing DBP2 and ICF02 fails. What happens for the lock and SCA structures follows this general process:

1. ICF02 fails.
1. The structure starts to rebuild in ICF01.
2. DBP1 contributes its part of each structure.
3. Then it is realized that DBP2 has not rebuilt its part of the structure.
4. Rebuild cannot complete, so there is fallback to the original structure, ICF02.
5. ICF02 is no longer there or accessible.
6. Because the rebuild cannot complete, DBP1's IRLM has no other option than to come down, and DBP1 will then fail. It must do this to protect data integrity. The same is true for the SCA. Loss of either one brings down the group for this condition.

It is important that the z/OS image with DBP2 be failed as soon as possible. Until that happens and the DBP2 status is "FAILED", no progress can be made. A z/OS image that is not responding can be quickly failed if the z/OS systems team implements a sysplex failure management (SFM) policy with ISOLATETIME(0) instead of PROMPT. While these are not settings that DB2 staff will be familiar with, if ISOLATETIME is not used, the failing image waits for an operator response to the prompt. Meanwhile the data sharing group can do nothing until the member's status is changed from "FAILING" to "FAILED".

Once DBP2 becomes FAILED, you can start DBP1 and it will recreate the lock and SCA structures in ICF01 by reading its own log and the logs of DBP2. It can do this even if DBP2 has not yet been started.

What has been happening to the GBPs in this scenario? If they are duplexed, they will fall back to simplex as soon as the DB2 has been marked FAILED, within seconds. They remain even though the group is down.

The scenario just described presents a group wide outage. There are two solutions to avoid this situation, other than *accepting the unlikely risk that an entire processor will fail*. In order to provide full availability against this condition, in the following sections we discuss two possible solutions:

- ▶ System-managed duplexing for DB2 lock and other structures
- ▶ External CF, such as z890 or z9 BC

### 3.4.2 System-managed duplexing for DB2 lock and other structures

System-managed duplexing can provide protection against a double failure. The duplexed lock structure and SCA will fall back to simplex, as the duplexed GBPs would do. As we have seen at 3.3.2, “System-managed duplexing” on page 55, none of this processing occurs without the cost of about 3 times or more than a simplexed lock access. It is probable that a conversion to asynchronous will occur for most locks. The asynchronous lock service time (as measured by the RMF CF activity report) is likely to be the predominate lock access time, and is significantly longer than synchronous access.

While the ICF itself is less expensive than an external CF, there are some additional costs for system-managed duplexing, the main one being the extra capacity in the host processors.

Consider this solution when your DB2 processing is a small part of your workload. It provides a simplified configuration with no stand-alone server.

### 3.4.3 External CF, such as z890 or z9 BC

An external CF in which the lock and SCA structures reside eliminates any double failure exposure.

Referring to Figure 3-8 on page 57, you can see that the external CF can either reflect configurations “B” or “D”. Configuration “D” requires a third footprint, but it is possible that there are multiple Parallel Sysplexes or multiple footprints, and that you can keep a DB2 member from our data sharing group off the processor with ICF02.

A moderately priced processor, such as the z890 or System z9 BC, can provide isolation from double failure. These processors have slightly slower internal speeds than the z990s and System z9 processors, but have many fewer engines (an S07 model can have up to 7 ICFs) and are cost-effective as external CFs.

The external CF needs two links from each processor, where the ICF-only configuration needs them only from the opposite processor (to the ICF). This configuration does need two (internal) IC links for the local processor, but there is no cost. An external CF avoids the extra host overhead incurred by the system-managed duplex solution. Access is likely synchronous to the lock structure when the technology generations are the same.

Links have different characteristics:

- ▶ ISC - Up to 100 km but slower
- ▶ ICB - Up to 7 m, but faster and more efficient

Failure is reduced to a single point (the external CF) and the lock and SCA structures now can rebuild into the ICF if the external CF fails. Loss of a processor does not affect the CF, which will remain intact.

**Important:** The highest availability is achieved with two external CFs. A practical compromise is one external CF (with Lock and SCA structures) and an ICF without duplexing for the Lock and SCA. Duplexing is considered mandatory for the GBPs as long as availability is a priority.

### Technology considerations for external CFs

An important factor to be considered for an external CF is the technology relative to the central processors. We recommend that there should be no more than a one generation difference between the processor and its external CF:

- ▶ A System z990 should have either a z990, z890, or z900 external CF.
- ▶ A System z9 should have either a System z9 BC or a z890 external CF.

The reason for such a recommendation is that the XCF is sensitive to the response time of the requests in relationship to the speed of the server it is running on. The response time depends on link speeds, CF, and distance to CF.

If there is more than one processor generation difference, it is likely that most CF requests will be converted to asynchronous dynamically by XES in order to spare the CPU cycles of the faster processor. Even if there is a difference of one generation, some of the CF requests that could have been synchronous will be converted to asynchronous. This means that requests will take a little longer and locks will be held longer than would be the case if the processor and CF technologies matched. Locks that remain synchronous will also take longer.

## 3.5 CFRM policy

There is one active coupling facility resource management (CFRM) policy per Parallel Sysplex. This policy contains descriptions of the coupling facilities available to it and the definitions of all the structures that can be allocated in the CFs. Statements similar to these will be incorporated by your z/OS Parallel Sysplex team into the existing CFRM policy.

Example 3-2 shows a snippet of the statements we have developed in this chapter.

*Example 3-2 Sample CFRM policy statements*

---

```
STRUCTURE NAME=groupname_GBP0
INITSIZE=16000
SIZE=32000
ALLOWAUTOALT(YES)
FULLTHRESHOLD=80
MINSIZE=16000
PREFLIST=(CF2,CF1)
DUPLEX(ENABLED)
REBUILDPERCENT(1)
```

We show only one GBP in this example for brevity.

```
STRUCTURE NAME=groupname_LOCK1
INITSIZE=64000
SIZE=76000
ALLOWAUTOALT(YES)
FULLTHRESHOLD=80
MINSIZE=64000
PREFLIST=(CF1,CF2)
REBUILDPERCENT(1)
```

```
STRUCTURE NAME=groupname_SCA
INITSIZE=64000
SIZE=128000
ALLOWAUTOALT(YES)
FULLTHRESHOLD=80
MINSIZE=64000
PREFLIST=(CF1,CF2)
REBUILDPERCENT(1)
```

---

The structure definitions shown in Example 3-2 were developed using sizing recommendations in this chapter:

- ▶ INITSIZE was developed according to sizing recommendations for each type of structure.
- ▶ SIZE was developed as roughly twice the size of the GBPs, assuming that Auto Alter was being implemented for capacity planning purposes as well as development of the directory to data ratios. We added a little extra storage to the lock structure in case the RLEs increase. We have allowed the SCA to increase as well.

- ▶ ALLOWAUTOALT(YES) was specified to implement Auto Alter as it pertains to each structure type.
- ▶ FULLTHRESHOLD allows Auto Alter to increase the structure size when the number of either entries or elements reaches 80% of each one. It is the default, but we think it is more straightforward to specify it.
- ▶ MINSIZE allows Auto Alter to decrease the structure size when the CF is under storage stress (<10% free CF storage) for structures that specify ALLOWAUTOALT(YES), but never to decrease it below this size. We have set it to the same value as INITSIZE for these critical production DB2 structures.
- ▶ PREFLIST orders the CFs where a structure should be placed and the preferred location for the structure. While we show two CFs, it is certainly possible to specify more than two. The secondary structure (CF2) is the one chosen for structure rebuild if the preferred structure becomes unavailable, either due to failure of the CF or the failure of the last link to the structure from a z/OS image containing a DB2 member of that group.
- ▶ We show the GBP0 in CF2 followed by CF1, and the reverse for the lock and SCA structures. Always place the lock and SCA structures in the same CF, according to 3.4, “CF configuration alternatives” on page 56. The GBPs do not have to be placed together, and usually are not. They are apportioned to the CFs based on load balancing for the total CF CPU utilization (>50% for a multi-engine CF and > 25% for single engine). Duplexed GBPs need no more storage than simplex because the storage in a 2-CF configuration must be available to handle the entire CF workload in failure situations.
- ▶ REBUILDPERCENT describes the percentage of loss of connectivity that is tolerated before XES rebuilds the structure. Because we always want the rebuild to occur to the other CF if any loss of connectivity is detected, we specify 1%.

Your z/OS systems team will know where to place all these structures.

For more formal definitions of these parameters, see *z/OS V1R7.0 MVS Setting Up a Sysplex*, SA22-7625-12.

## 3.6 Best practices for the coupling facility

Here is a summary of our recommendations for best practices for the coupling facility:

- ▶ Use CFSizer to size the GBPs.
- ▶ Start with 64000 for lock structure size.
- ▶ Start with 64000 for SCA structure size.
- ▶ Implement Auto Alter for the GBPs as long as enough storage exists in the CFs for the GBPs to be correctly sized.
- ▶ Make sure one CF can take over the workload in both CFs in terms of CF storage.
- ▶ Ensure that all DB2 structures rebuild into another CF upon any failure.
- ▶ Duplex the GBPs specifying DUPLEX (ENABLED). Because GBP-dependency occurs only at the time data is shared, the ALLOWED parameter requires operator intervention.
- ▶ Guard against double failure in a Parallel Sysplex through use of an external CF for the Lock and SCA structures and an ICF.
- ▶ Provide 2 links from each image to each CF.
- ▶ Match an external CF's technology to the processor.
- ▶ Provide 2 dedicated engines minimum per CF for a production Parallel Sysplex. Add another, when peak CF CPU utilization exceeds 50%
- ▶ If each CF is a single engine, add a second one when peak CF CPU utilization approaches 25%.
- ▶ Set GBP thresholds:
  - GBP Checkpoint 4 minutes
  - GBPOOLT 10-30%
  - CLASST 2-5%







## Implementing data sharing

In this chapter we examine several important topics that you must address when you implement DB2 data sharing:

- ▶ Naming conventions
- ▶ Logging
- ▶ DSNZPARMs
- ▶ Renaming an existing non-data sharing member
- ▶ Enabling the data sharing group
- ▶ Adding the second member
- ▶ Removing a member

In rare cases, you may find it necessary to disable a data sharing group and later re-enable data sharing. The processes are provided in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-741.

## 4.1 Naming conventions

You should carefully consider the naming convention you define for your DB2 data sharing group. Using a consistent naming convention will help you identify which entities belong to a particular DB2 member, even if that member is moved to another LPAR in the Parallel Sysplex.

In this section we discuss the following topics:

- ▶ Group name
- ▶ Group attachment name
- ▶ Subsystem identifier (SSID)
- ▶ Log data set names
- ▶ Bootstrap data set (BSDS)
- ▶ Distributed Data Facility (DDF) related parameters
- ▶ Catalog alias
- ▶ Temporary work file database
- ▶ Some more naming recommendations
- ▶ Examples of naming conventions
- ▶ Best practices for naming conventions

### 4.1.1 Group name

The group name is the name that represents the entire data sharing group to cross-system coupling facility (XCF).

The data sharing group name identifies this group to the Parallel Sysplex. The data sharing group name is specified primarily for Coupling Facility Resource Management (CFRM) policy purposes and for related XCF commands. The coupling facility structure names are based on this name using the following convention: *groupname\_LOCK1*, *groupname\_SCA*, *groupname\_GBP0*, and so forth. See Table 4-1.

There are a few restrictions on the first letter of each name. Here is an excerpt of these restrictions, taken from the section “Data sharing group names” in Chapter 2, “Planning for DB2 data sharing”, of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417-02.

**Restrictions:** To avoid names that IBM uses for its XCF groups, do not begin DB2 group names with the letters A-I unless the first three characters are DSN. Do not use the string SYS as the first three characters, and do not use the string UNDESIG as the group name.

**Important:** Never reuse a DB2 group name, even if a data sharing group that previously used the name no longer exists. Some data sharing information, such as the DB2 group name, is retained in the Parallel Sysplex couple data set (CDS).

For example, we have seen names such as DSNPROD, DSNPRD1, DSNPRD2, and DSNTST1 used as group names. However, DB2PRD1 cannot be a group name, because it violates the stated restrictions.

## 4.1.2 Group attachment name

The group attachment name acts as a generic name for all the members of the data sharing group. Utilities and applications that use TSO, batch, DL/I batch, the RRSAP, the CAF, or DB2I can connect to any member in the group by using the group attachment name. This allows you to run these applications on any member without making any changes to the JCL.

Here are our recommendations for developing a naming convention:

- ▶ Make the group attachment name the SSID of the originating member of the data sharing group. The existing JCL already has this value specified, so no changes to your procedures will be required.
- ▶ Rename your existing SSID according to your new naming convention. There are instructions for doing so in the section “Renaming a member” in Chapter 3, “Installing and enabling DB2 data sharing” of *DB2 for z/OS Version 8 Data sharing: Planning and Administration*, SC18-7417. We cover this process in 4.4, “Renaming an existing non-data sharing member” on page 78.
- ▶ Do not name your SSID the same as your group attach even if the name fits with your naming convention. This becomes a consideration when multiple members of your data sharing group are running on the same z/OS image.
  - As long as the SSID is specified in the JCL or the CICS AOR, for example, the connections will be made to the specific member.
  - If a group attach is specified as SSID, then the connection is made to the *first* DB2 active member on that z/OS image, according to the IEFSSNxx member of SYS1.PARMLIB. No balancing of the workload occurs. For more information on subsystem search order, refer to the section “Specifying the group attachment name” in Chapter 2, “Planning for data sharing” of *DB2 for z/OS Version 8 Data sharing: Planning and Administration*, SC18-7417.
  - Some large DB2 data sharing groups might run two members of the group on the same z/OS image. DB2 provides a way to balance the workload between the two members, as described in 7.7, “Usermod for routing to

multiple DB2s on a z/OS image” on page 130. This method cannot be used if there is an actual member name that is the same as the group attach name.

For more information, see Chapter 2, “Planning for DB2 data sharing” of *DB2 for z/OS Version 8 Data sharing: Planning and Administration*, SC18-7417. We also provide some specific examples in Table 4-1 on page 72.

### 4.1.3 Subsystem identifier (SSID)

A SSID naming convention in a data sharing environment should provide for easy identification of members within a data sharing group and automation routines. For example, members could be DBP $x$ , where  $x$  represents a value of 1-9 and/or A-Z. Naming the IRLM SSID should be consistent as well. See Table 4-1 on page 72 for examples.

Using SSIDs that do not follow a naming convention can cause confusion for operations and increase complexity for automation routines. An example of this might be: ABCD, DBP1, DBP2.

### 4.1.4 Log data set names

Each member in the data sharing group still has its own pairs of active logs and archive logs. To easily identify the log data sets that belong to a particular member, we recommend that the SSID be included as part of the data set name. In addition, the DSNZPARM `TIMESTAMP=YES` makes it much easier to identify archives necessary for problem diagnostic purposes. To avoid exceeding the MVS limit of 44-character data set names, we suggest that the archive log data set name be specified as shown in the second line in Example 4-1.

*Example 4-1 Active and archive log names*

---

```
groupname.ssid.LOGCOPY1.DS01 → DSNDB2P.DBP1.LOGCOPY1.DS01  
groupname.ssid.ARCHLOG1 → DSNDB2P.DBP1.ARCLG1.Ddyddd.Thmmsst.Axxxxxxx
```

---

### 4.1.5 Bootstrap data set (BSDS)

The BSDS of each member contains information about all the other members in the data sharing group. This is done so that any DB2 member can read any other member's BSDS, active logs and archive logs. For example, when one member is running the RECOVER utility for a DB2 object, it may need to read other members' BSDSs, active logs, and archive logs. Make sure that the two copies of the BSDS are allocated on different disk subsystems for high availability.

We recommend that the SSID be included as part of the data set names. See Example 4-2.

*Example 4-2 BSDS naming convention*

---

groupname.ssid.BSDS01	→	DSNDB2P.DBP1.BSDS01
groupname.ssid.BSDS02	→	DSNDB2P.DBP1.BSDS02

---

## 4.1.6 Distributed Data Facility (DDF) related parameters

There are several DDF parameters that must be defined when you plan to implement data sharing. These include DB2 LOCATION NAME, DB2 NETWORK LUNAMES, and a DB2 GENERIC LUNAME. In the case of TCP/IP, you need to add DRDA® PORT and RESYNC PORT.

### ***DB2 location name***

The data sharing group is treated as a single location by remote requestors. DB2 location name is the name that encompasses the entire data sharing group. An example might be DB2PLOC. It is defined in install panel DSNTIPR and it is the same for every member of the data sharing group.

### ***Location aliases***

Location aliases represent one member, a few members, or all members of a data sharing group. For TCP/IP connections, you can additionally specify an alias-port to indicate that a member belongs to a subset of the group. For system network architecture (SNA) protocol, you can define one or more alias-names using the ALIAS option of the change log inventory utility (DSNJU003) for the location of the data sharing group. For more information about the ALIAS option, see Chapter 36, “DSNJU003 (change log inventory)” of *DB2 UDB for z/OS Version 8 Utility Guide and Reference*, SC18-7427. Location aliases provide the following benefits:

- ▶ Location aliases allow you to define subsets of your data sharing group. This gives you the ability to control the members to which remote requesters can connect.
- ▶ As you enable an existing non-data sharing subsystem to data sharing or merge a DB2 system to a data sharing group, you can use the DB2's old location name as a location alias to point to the location name of the data sharing group. No application changes are required.

For more information about location aliases, see Chapter 4, “Communicating with data sharing groups” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

### ***DB2 network logical unit name (LU name)***

This name identifies a DB2 subsystem to VTAM® uniquely within the group and the network. Even if you are planning to use only TCP/IP to communicate with remote sites, you still have to configure the LU name parameter and VTAM because DB2 uses the network ID and the LU name to identify units of work. You should use a convention similar to that used for the SSID. An example is LUDBP1 for DBP1.

### ***DB2 generic LU name***

If you use RACF® pass tickets for your security, use the generic LU name for RACF to make a valid pass ticket for the remote requesters as a single LU name for the data sharing group. At this point, the remote requesters could have their own LU name or generic LU name.

If you do not use both RACF pass ticket and group-generic access, then ignore the generic name and leave it blank.

We recommend member-specific access instead of group generic for workload balancing.

### ***DRDA port***

If you use TCP/IP network connections, you must specify the DRDA listener port for the data sharing group. Each member should have the same value for this DRDA port. Your TCP/IP systems administrator will assign the port numbers to use. Port 446 is the common SQL listener port, but obviously it can only be used for one data sharing group. It is defined in install panel DSNTIPS.

### ***RESYNC port***

The RESYNC port is used to process requests for two-phase commit re-synchronization. The number for this port must be different from the DRDA listener port. Each member has its own unique numbers within the Parallel Sysplex. Your TCP/IP systems administrator will assign the port numbers to use.

### ***Parallel sysplex domain name***

This name lets you take advantage of workload balancing and two-phase commit for TCP/IP connections. If your site uses a domain name server (DNS), you must register this name (in the format *location.sysplex.domainname*) with the server.

The sysplex name is recorded in the COUPLExx z/OS data set. The domain name comes from the socket calls, *gethostid* for the host address, and *gethostbyaddr* for the host name. For more information about the Parallel Sysplex domain name, see “Registering names in the domain name server” in Chapter 4, “Communicating with data sharing groups” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

### ***Member domain name***

DB2 handles indoubt thread resolution for TCP/IP connections through this name. You should register each member's domain name (in the format *luname.location.sysplex.domainname*) with the domain name server. For more information about member domain name, see "Registering names in the domain name server" in Chapter 4, "Communicating with data sharing groups" of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

## **4.1.7 Catalog alias**

The catalog alias is used to indicate the ICF catalog used for DB2 VSAM data sets such as DSNDB01, DSNDB06, the logs, and BSDS. If you do not plan to use the BACKUP SYSTEM / RESTORE SYSTEM DB2 online utilities, you can use the group name as a catalog alias.

If you plan to use the BACKUP SYSTEM DB2 utility, you must have different ICF catalogs for the DB2 active logs and BSDS data sets from those for the Catalog/Directory and user database. The two groups are in separate DFSMS *copy pools* and are backed up separately. For restoration they must be in sync with the data they represent at the same point-in-time. For more information about copy pools, see *z/OS DFSMSdfp™ Storage Administration Reference*, SC26-7402. And for more information about BACKUP SYSTEM utility and RESTORE SYSTEM utility, see *DB2 UDB for z/OS Version 8 Utility Guide and Reference*, SC18-7427.

## **4.1.8 Temporary work file database**

A DB2 subsystem in non-data sharing uses DSNDB07 for the temporary workfile database. In a data sharing environment, each member has its own work file database. It is important to have a meaningful name, such as WRKDBP1, so you can easily identify which one belongs to each DB2 system.

## **4.1.9 Some more naming recommendations**

In the following sections, we describe a few more naming recommendations.

### ***Command prefix***

We recommend that you prefix a special character to the subsystem identifier, such as the default -DBP1.

### ***Load module for subsystem parameters***

We recommend that you include a subsystem identifier in the load module, for example, DSNZDBP1.

### ***IRLM group name***

Your IRLM subsystems join an XCF group. We recommend that you include your data sharing group name as part of the IRLM group name such as DXRDB2P. For more information regarding the IRLM group name, see “IRLM names” in Chapter 2. “Planning for DB2 data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

### ***IRLM subsystem name***

We recommend that you have the IRLM subsystem name paired with its DB2 subsystem name, such as IRP1 for DBP1.

## **4.1.10 Examples of naming conventions**

Table 4-1 shows examples of DB2 related entities for DB2 data sharing. If you are not performing a new installation, you will have to keep some of the entity names from your existing DB2 subsystem for which you are going to enable data sharing.

*Table 4-1 Examples for naming conventions*

Entity	Example
z/OS system name	SYSA, SYSB
DB2 group name	DSNDB2P
DB2 group attach name	<b>DB2P</b> use your existing DB2 SSID
DB2 subsystem name	<b>DBP1, DBP2</b>
Command prefix	<b>-DBP1, -DBP2</b>
IRLM subsystem name	IRP1, IRP2
IRLM XES group name	DXRDB2P
Catalog alias	DSNDB2P you can keep your existing DB2 SSID alias <b>Important:</b> If you plan to use BACKUP SYSTEM and RESTORE SYSTEM DB2 online utility, you should have one ICF catalog for logs and BSDSs and another ICF catalog for databases.
System load module name	DSNZDBP1, DSNZDBP2
Temporary work file name	WRKDBP1, WRKDBP2
DB2 location	<b>DB2PLOC</b> - use your existing DB2 SSID location
DB2 generic LU name	<b>LUDB2P</b> - use your existing DB2SSID LUNAME



Entity	Example
DB2 network LU name	LU <b>DBP1</b> , LU <b>DBP2</b>
BSDS name	DSNDB2P. <b>DBP1</b> .BSDS01 DSNDB2P. <b>DBP1</b> .BSDS02
Log data set name	DSNDB2P. <b>DBP1</b> .LOGCOPY1.DS01 DSNDB2P. <b>DBP1</b> .ARCLG1.Dyyddd.Thhmsst.Axxxxxxx

### 4.1.11 Best practices for naming conventions

Here are our recommendations for choosing a naming convention:

- ▶ Keep the DB2 subsystem name of your originating non-data sharing member as your group attach name, to avoid making changes to your applications.
- ▶ Do not name any SSID the same as your group attach name.
- ▶ Include the SSID in the BSDS and log data set names for each member.

## 4.2 Logging

Each DB2 member has its own active and archive log data sets.

In this section we discuss the following topics:

- ▶ Active log data sets
- ▶ Archive log
- ▶ Best practices for DB2 logging

### 4.2.1 Active log data sets

As with non-data sharing, we recommend keeping the active log copy 1 for each member on a separate disk subsystem from that of the corresponding active log copy 2 to assure data integrity and availability. It is possible for any disk to fail. If its data should not be recoverable, and both copies of the current active log are co-located, there would be broken DB2 data with no way to determine the extent of the damage. Similar considerations apply to BSDS.

We recommend that each member should have enough disk space for active and archive data sets such that tape archive backup would not have to be accessed during the recovery of a critical DB2 object. Critical data is usually image-copied every 24 hours, so each member should have at least 24 hours worth of data available on disk.

## 4.2.2 Archive log

We recommend archiving to disk. Because the recovery operation may have to read and merge the logs of several members, the recovery process is optimized when all log data is on disk.

When the archives are migrated from disk to tape, make sure that the archives for each member do not end up on the same tape. Two different DB2 subsystems may call for their archives at the same time period and one of them would have to wait, or possibly they would deadlock with each other.

If you archive directly to tape devices, make sure that you have enough tape devices to allocate all members' archive logs at the same time to improve recovery performance.

## 4.2.3 Best practices for DB2 logging

We recommend the following best practices for DB2 logging:

- ▶ Make sure that active log copy 1 is on a different disk subsystem from that of active log copy 2.
- ▶ Archive to disk.

## 4.3 DSNZPARMs

During your planning process, you should become familiar with certain data sharing related system parameters (DSNZPARMs). Also, there are some parameters in the IRLM startup procedure that you should change, and some new information that is stored in the BSDS.

In this section we discuss the following topics:

- ▶ Data sharing system parameters
- ▶ Some other important DSNZPARMs
- ▶ IRLM parameters
- ▶ Information stored in the BSDS

### 4.3.1 Data sharing system parameters

Table 4-2 shows the system parameters that you should check when you enable data sharing or add members to a data sharing group.

Table 4-2 Data sharing related system parameters

Parameter	Macro	Panel	Remark
ASSIST	DSN6GRP	DSNTIPK	This parameter allows this member to assist in sysplex query parallel processing. You should specify VPPSEQT and VPXPSEQT buffer pool thresholds of this member to be greater than zero.
COORDNTR	DSN6GRP	DSNTIPK	This parameter allows this DB2 member to be the coordinator for sysplex query parallel processing with other members of the group.
DSHARE	DSN6GRP	DSNTIPA1	This parameter indicates whether data sharing has been enabled on the subsystem. Use YES in a data sharing environment.
GRPNAME	DSN6GRP	DSNTIPK	See 4.1.1, “Group name” on page 66.
MEMBNAME	DSN6GRP	DSNTIPK	See 4.1.3, “Subsystem identifier (SSID)” on page 68”
RETLWAIT	DSN6SPRM	DSNTIPI	Keep the default of zero. With reduced IRLMRWT values, it is hard to restart a DB2 within a reasonable window.
SSID	DSNHDECP	DSNTIPM	See 4.1.2, “Group attachment name” on page 67

### 4.3.2 Some other important DSNZPARMs

Here we describe some DSNZPARMs you need to understand:

<b>CTHREAD</b>	There is nothing inherently different for this parameter from its use in non-data sharing. It is one of the key values affecting the virtual storage consumption of the DBM1 address space. See 7.8, “Determining the number of threads” on page 131 for a discussion on how this number should be chosen.
<b>MAXDBAT</b>	There is nothing inherently different for this parameter from its use in non-data sharing. It is one of the key values affecting the virtual storage consumption of the DBM1 address space. See 7.8, “Determining the number of threads” on page 131 for a discussion on how this number should be chosen.
<b>LOGAPSTG</b>	The only change you encounter for data sharing recovery is for GREP/LPL activity, when it occurs during restart in DB2 V8. The default value is 100MB and should be set for best performance for GRECP/LPL recovery (as well as for the RECOVER utility).

<b>SYNCVAL</b>	There is no difference that occurs in data sharing except the need to perform analysis on all members or a group. When they have differing statistics time starts, it is difficult to match them with each other or with an RMF interval that the z/OS systems staff might have. Specify it at the offset to an hour that the z/OS systems staff uses for RMF. Commonly it is 0,15,30,45 (for 15 minute intervals), but occasionally an installation will choose differently.
<b>PCLOSEN</b>	For this setting and that for PCLOSET, we recommend that you look at “Data sets converted from R/W → R/O” in the Statistics long report. Leave the default until you see more than 10-20 per minute converted to R/O. See 7.1, “CLOSE YES and CLOSE NO table spaces” on page 122.
<b>PCLOSET</b>	For this setting and that for PCLOSEN, we recommend that you look at “Data sets converted from R/W → R/O” in the Statistics long report. Leave the default until you see more than 10-20 per minute converted to R/O. See 7.1, “CLOSE YES and CLOSE NO table spaces” on page 122.
<b>CHKFREQ</b>	We recommend 2 or 3 minutes for System z processors and for each member. There are certain values for data sharing that need to keep advancing. Having one relatively inactive member whose CHKFREQ is based on number of records would delay checkpointing for too long.
<b>URCHKTH</b>	This value is just as important in non-data sharing, but the impact of a runaway updating job without commits has wider scope. See 7.6, “The problem of runaway threads” on page 128.
<b>URLGWTH</b>	This value is just as important in non-data sharing, but the impact of a runaway updating job without commits has wider scope. See 7.6, “The problem of runaway threads” on page 128.
<b>LRDRTHLD</b>	While this value affects only readers, it is well paired with URLGWTH and URCHKTH as it can prevent the Switch phase of Online REORG utility. See 7.6, “The problem of runaway threads” on page 128.

### 4.3.3 IRLM parameters

Table 4-3 shows the IRLM parameters that you should check when you enable data sharing or add members to a data sharing group. The installation CLIST panels creates a procedure with correct data sharing values set.

Table 4-3 Data sharing related with IRLM parameters in IRLM start procedure

Parameter / Field name	Macro	Remark
Scope / Disconnect IRLM	DSNTIPJ	Specify GLOBAL
IRLMGRP / IRLM XCF group name	N/A	Do not start the group name with SYS and do not use UNDESIG. Use a name starting with DXR if it is possible.
LOCKTAB / Lock entry size	DSNTIPJ	Keep default of 2 for width of lock table. It matches with MAXUSRS=7.
MAXUSRS / n/a	N/A	Specify 7
IRLMID / Member identifier	DSNTIPJ	
LTE / Number of lock entries	DSNTIPJ	You can specify the number of lock table entries using this parameter. If you do not, IRLM determines the number of lock table entries based on the lock structure size that is specified in CFRM policy and the number of users (MAXUSRS). For more information about LTE, see 3.1.2, "Sizing the lock structure" on page 46 and Chapter 2."Planning for DB2 data sharing" of <i>DB2 for z/OS Version 8 Data Sharing: Planning and Administration</i> , SC18-7417.

### 4.3.4 Information stored in the BSDS

Table 4-4 shows the BSDS information that is provided as output of the installation CLIST when you enable data sharing or add members to a data sharing group. For more information, see Chapter 6, "Installing, migrating, and updating system parameters" of *DB2 UDB for z/OS Version 8 Installation Guide*, GC18-7418. For more information regarding change log inventory job, see Chapter 36. "DSNJU003 (change log inventory)" of *DB2 for z/OS Version 8 Utility Guide and Reference*, SC18-7427.

Table 4-4 Data sharing input to change log inventory

Field panel name	Panel	Remark
DB2 location name	DSNTIPR	This will apply to the entire data sharing group.
Location aliases	n/a	This represents one, more or all members of your data sharing group.
DB2 network LU name	DSNTIPR	Each DB2 member has its own LU name.

Field panel name	Panel	Remark
Generic LU name	DSNTIPR	Required If you want RACF pass tickets. Required for group-generic access using SNA (not recommended).
DRDA port	DSNTIP5	Each DB2 has the same value.
RESYNC port	DSNTIP5	Each DB2 has its own value.

## 4.4 Renaming an existing non-data sharing member

As stated earlier, defining and adhering to a naming standard is especially important. We recommended that the SSID of the existing non-data sharing subsystem that will become the originating member of the data sharing group should become the group attachment name. We also recommended that the SSID of every member should be different from the group attachment name. Use this procedure to change the SSID of the originating member to a new name based on your naming standards.

The process for renaming a DB2 subsystem is given in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, under “Renaming a Member”. We will refer to steps within its sections “Tasks that require an IPL” and “Tasks at enable time”. We want you to follow those steps, but we elaborate on some of them, respectively, in 4.4.1, “IPL required for these tasks” on page 79 and 4.4.2, “Enable data sharing tasks” on page 79 of this book and provide additional information.

The rename process assumes that you take one outage at the time you enable data sharing to rename your DB2 subsystem from the old name to the new name. Naturally we want you to make the outage as short as possible. In this section, as an example of the old name, we will use ABCD. For the new names, we will use DBP1 for the first member and DBP2 for the second member.

**Note:** The renaming procedure does not change any high level qualifiers, but renames certain data sets.

It is not necessary to change the high level qualifier of any of these data sets or the DB2 data. If you do decide to change it, we recommend that you do so separately from this procedure, due to its complexity and the outage it incurs. The section in the *DB2 for z/OS Version 8 Administration Guide*, SC18-7413, called “Changing the high-level qualifier for DB2 data sets” contains a procedure to accomplish this task.

### 4.4.1 IPL required for these tasks

After you have chosen your new naming convention, you can perform the “Tasks that require an IPL” at any time. The only activity is the need to update the IEFSSNxx member of SYS1.PARMLIB. You must:

1. Update the existing subsystem entry to add the group attach name. (If you do not do it exactly as coded in the example in the manual, jobs using the call attach facility (CAF) may not work correctly.)
2. Add the subsystem entries for *both* the data sharing members you will implement (according to the naming convention), in our case, DBP1 and DBP2.

While you can use the z/OS SETSSI command to *add* new IEFSSNxx statements without an IPL, you cannot use it to *update* an existing subsystem. Therefore, plan for an IPL well ahead of the time you are planning to start DB2 in data sharing mode.

### 4.4.2 Enable data sharing tasks

Perform step 1 and step 2 as described in the section “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

Part of this section in the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, instructs you to invoke the installation panels to enable data sharing with the parameters specific to data sharing. Because you cannot change the DB2 subsystem name with the installation panels, the output PDS members of a NEW.SDSNSAMP data set will have data sharing specific information, but will have the old (ABCD) SSID. See “Changing the rest of the members of SDSNSAMP PDS” on page 86.

#### Hints for the installation panel specifications

You invoke the installation CLIST in step 3 of “Tasks at enable time”. Stop before step 3 and continue here: The hints given in this section provide more specific information on this topic.

1. On the command line of the first panel, specify: **panel id**.

You are now on panel DSNTIPA1.

- a. Here you specify these values:

Install Type: install

Data Sharing: yes

You then see a pop-up, panel DSNIPP.

- b. Choose “3” to enable data sharing.

2. Panel DSNIPK is where you enter your values for these parameters:

GROUP NAME: **DSNDB2P**  
MEMBER NAME: **DBP1**  
WORK FILE DB name: **WRKDBP1**  
GROUP ATTACH: **ABCD**  
COORDINATOR: enter your value  
ASSISTANT: enter your value

As the manual states, you should specify the old subsystem name, ABCD, as the group attach name. Specifying the new member name of DBP1 ensures that the DSNTIJUZ job is generated correctly, as is the DSNTIJTM that creates a work data base for this subsystem. The “Member Name” is not the same as the SSID, as you will see in a later panel.

3. On panel DSNTIPH, specify:

COPY 1 PREFIX: *hlq*.DBP1.ARCLG1  
COPY 2 PREFIX: *hlq*.DBP1.ARCLG2  
TIMESTAMP ARCHIVES: yes

4. On panel DSNTIPT, specify the output data set that the installation CLIST will allocate and in which it will place the members it tailors. It is helpful to have the new member name as part of the data set name, which will be useful when you have multiple members of the group;

SAMPLE LIBRARY: *hlq*.DBP1.SDSNSAMP

**Note:** Make sure you enter a different data set for your SAMPLE LIBRARY or you will overlay the one you are currently using.

5. On Panel DSNTIPO, enter the name of our DSNZPARM module using the new member name, **DSNZDBP1**.

6. On panel DSNTIPI, specify:

INSTALL IRLM: YES  
SUBSYSTEM NAME: **IRP1**  
PROC NAME: **DBP1IRLM**

7. Panel DSNIPJ is pre-filled for the first IRLM in a group. The only value you enter is:

IRLM XCF GROUP NAME: **DXRDB2P**

8. On panel DSNTIPM, the PARMLIB updates panel, you cannot change the subsystem name (it is “greyed out”). We suggest you leave the “Command Prefix” as it is, because you have already updated the IEFSSNxx member with the correct values. (The command prefix is not stored in DSNZPARM).



9. Panel DSNTIPR contains the DDF information:

DB2 LOCATION NAME: **LOCABCD**  
DB2 NETWORK NAME: **LUDBP1**  
DB2 GENERIC LUNAME: **LUABCD**

The location name of the existing DB2 becomes the group location name. The *luname* of the renamed subsystem is LUDBP1. The luname of the existing DB2 becomes the generic luname.

Later you can establish an alias for the group location name if you do not want to use LOCABCD.

Most of the installation panels are presented as they would have been during an initial installation of DB2, but with the values you cannot change “greyed out”, such as the active logs and the BSDS data sets. There are numerous members in your newly customized SDSNSAMP data sets, though not quite as many as those from an initial installation. See “Changing the rest of the members of SDSNSAMP PDS” on page 86.

## Remaining steps of the rename process

Here is the remainder of our renaming procedure:

1. You stop DB2 in Step 4 of “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417. Ensure that all activity is complete with these commands:
  - STOP DB2 MODE(QUIESCE)
  - START DB2 ACCESS(MAINT)
  - **DB2 status check**
    - DISPLAY THREAD(\*)
    - DISPLAY UTILITY(\*)
    - DISPLAY DB(\*) SPACENAM(\*) RES LIMIT(\*)
    - STOP DB2 MODE(QUIESCE)
2. Before you submit the DSNTIJUZ job during Step 5 of the “Tasks at enable time”, verify the DSN6GRP MEMBNAME=DBP1, the new subsystem name after the rename procedure is complete. Then run the DSNTIJUZ job to create the DSNZDBP1 DSNZPARM and the DSNHDECP member (of which there is only one for the group) in *hlq.DSNDB2P.SDSNEXIT*. Because SDSNEXIT is used for the entire group, we chose to specify the group name as the middle qualifier.

DSNTIJUZ changes the DDF information you entered in panel DSNTIPR, and a snippet of that job is shown in Example 4-3.

*Example 4-3 Change Log Inventory from DSNTIJUZ for DDF*

---

```
//*****  
//*          CHANGE LOG INVENTORY:  
//*          UPDATE BSDS  
//*****  
//DSNTLOG EXEC PGM=DSNJU003,COND=(4,LT)  
//STEPLIB DD DISP=SHR,DSN=DSN810.SDSNLOAD  
//SYSUT1 DD DISP=OLD,DSN=DB2810.BSDS01  
//SYSUT2 DD DISP=OLD,DSN=DB2810.BSDS02  
//SYSPRINT DD SYSOUT=*  
//SYSUDUMP DD SYSOUT=*  
//SYSIN DD *  
DDF LOCATION=LOCABCD,LUNAME=LUDBP1,  
GENERIC=LUABCD,RESPORT=5447,PORT=5446  
DDATASHR ENABLE
```

---

3. Next, submit the DSNTIJIN job that changes active logs to SHAREOPTIONS(2,3). The data set names in the JCL are for the active logs *before* you have renamed them.
4. Perform step 6 of “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.
5. Perform step 7 of “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 with these additions.
  - a. In Step 7a, write down the starting and ending timestamps in addition to the starting and ending ranges for each active log from the print log map utility (DSNUJ004). The current active log pair will have the status of NOTREUSABLE. The current active log pair when you stopped DB2 will not have an ending timestamp, because the data set has not been completely filled.
  - b. Perform steps 7b and 7c.
  - c. In step 7d when you run change log inventory utility (DSNUJ003) you use the NEWLOG ADD control statement for each renamed active log data, along with its original range and timestamps. Because you have no ending timestamp for the current active log pair, you can substitute the current time (*now*). When DB2 is later started and the log fills, it will update this value.
6. Perform step 8 and 9 of “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 if necessary.
7. Step 10 is the rename of the startup procedures. Your new DSNTIJMV job has two cataloged procedures to be copied to SYS1.PROCLIB: ABCDMSTR, and DBP1IRLM.

See Example 4-4 for the DB2 master address space started task procedure.

*Example 4-4 DSN TIJMV for MSTR started task*

---

```
./ ADD NAME=ABCDMSTR
/*****
/*      JCL FOR PROCEDURE FOR THE STARTUP
/*      OF THE DB2 CONTROL ADDRESS SPACE.
/*
/*      INSTALLATION MAY CHANGE PROGRAM LIBRARY
/*      NAMES IN STEPLIB DD STATEMENT TO THE
/*      LIBRARY IN WHICH DB2 MODULES ARE
/*      LOADED USING THE PROCEDURE VARIABLE:
/*      LIB
/*
/*****
//JUDYMSTR PROC LIB='DSN810.SDSNEXIT'
//IEFPROC EXEC PGM=DSNYASCP,DYNAMNBR=119,REGION=OK,
//      PARM='ZPARM(DSNZDBP1),GROUP(DSNDB2P),MEMBER(DBP1)'
//STEPLIB DD DSN=&LIB,DISP=SHR
//      DD DISP=SHR,DSN=DSN810.SDSNLOAD
//BSDS1   DD DISP=SHR,DSN=hlq.BSDS01
//BSDS2   DD DISP=SHR,DSN=hlq.BSDS02
```

---

The PARM parameters are correct, because you entered them on the installation panel. You must change the name of the procedure to DBP1MSTR (it kept the old name because you were not able to change the SSID through the installation CLIST.) You must change the BSDS1/BSDS2 DD statements to have DBP1 as the middle qualifier, just as you did when they were renamed in step 7 in “Tasks at enable time”.

The other cataloged procedure is shown in Example 4-5.

*Example 4-5 IRLM cataloged procedure*

---

```
./ ADD NAME=DBP1IRLM
/*****
/*      JCL FOR PROCEDURE FOR THE STARTUP
/*      OF THE IRLM 2.2 LOCK MANAGER
/*      ADDRESS SPACE.
lines deleted ....
/*****
//DBP1IRLM PROC RGN=5000K,
//      LIB='DSN810.SDXRRESL',
//      IRLNM=IRP1,
//      IRLMID=1,
```

```

//          SCOPE=GLOBAL,
//          DEADLOK='1,1',
//          MAXCSA=0,
//          PC=YES,
//          MAXUSRS=7,
//          IRLMGRP=DXRDB2P,
//          LOCKTAB=,
//          TRACE=YES,
//          PGPROT=YES,
//          LTE=0,
//          MLMT=2G
//          EXEC PGM=DXRRLM00,DPRTY=(15,15),
//          PARM=(&IRLMNM,&IRLMID,&SCOPE,&DEADLOK,&MAXCSA,&PC,
//              &MAXUSRS,&IRLMGRP,&LOCKTAB,&TRACE,&PGPROT,&LTE),
//          REGION=&RGN,
//          MEMLIMIT=&MLMT
//STEPLIB DD DSN=&LIB,DISP=SHR
//*
/* The following DUMP DD card should not be specified unless you
/* are having IRLM STARTUP problems and are not getting the dump
/* needed to diagnose the problem.
/*SYSABEND DD SYSOUT=*
/*SYSUDUMP DD  SYSOUT=*
/*DSNUPROC PEND          REMOVE * FOR USE AS INSTREAM PROCEDURE
./  ENDUP

```

---

This procedure is correct as it is and only needs to be copied to SYS1.PROCLIB. The parameters that are bolded were created by the installation CLIST. Of course, you supplied the IRLM group name of **DXRDB2P** and the procedure name, **DBP1IRLM**.

The other procedures for address spaces that you need to rename manually are:

- ▶ ABCDDBM1 > DBP1DBM1
- ▶ ABCDDIST > DBP1DIST
- ▶ ABCDSPAS > DBP1SPAS

If you have any WLM environments prefixed with ABCD, you do not need to rename them. You do, however, have to change the WLM environment for the WLM DB2SSN parameter &IWMSSNM. This parameter tells WLM to substitute the SSID of the DB2 on that z/OS image. Therefore, you can have one cataloged procedure in PROCLIB for an environment that all DB2s in the group can use. See 5.2, “Workload Manager” on page 92.

8. Complete the steps 11 and 12 of the section “Tasks at enable time” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, then continue as described in this book in 4.5, “Enabling the data sharing group” on page 85 without executing step 13.

## 4.5 Enabling the data sharing group

After renaming the existing non-data sharing subsystem, you can enable DBP1 for data sharing as an originating member. For more information, see “Enabling DB2 data sharing” in Chapter 3. “Installing and enabling DB2 data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417”.

After you finish the installation panels described in Chapter 3. “Installing and enabling DB2 data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, you need the following four jobs, at a minimum, to enable data sharing:

DSNTIJMV, DSNTIJIN, DSNTIJUZ, DSNTIJTM

### ***DSNTIJMV***

The contents of this member has new copies of the started procedures for the DB2 and IRLM address spaces, with the data sharing specific parameters that you entered on the installation CLIST ISPF panels.

During the rename process, you copied those members to PROCLIB with the modifications.

### ***DSNTIJIN***

DSNTIJIN is run to allow the other DB2 members of your data sharing group to have read access to this member’s active log data sets. It changes your active logs to have SHAREOPTIONS (2,3).

You have already run this job as part of the rename process.

### ***DSNTIJUZ***

You have already run this job. Its name has the new member name and the SSID within DSNTIJUZ is correct for the new name.

### ***DSNTIJTM***

This job creates a temporary work file database (described later).

### ***Changing the rest of the members of SDSNSAMP PDS***

Change each of the PDS members of your SDSNSAMP data set that have the SSID specified as ABCD. When you make your current SSID, ABCD, the group attach name, there are far fewer changes to be made. As an example, you can specify in the **DSN SYSTEM(ssss)** command statement, the group attach name, as well as the SSID invoked for the utilities that are run in SDSNSAMP.

We illustrate this concept in Example 4-6.

*Example 4-6 Snippet of sample job DSNTEJ1*

---

```
//PH01S06 EXEC PGM=IKJEFT01,DYNAMNBR=20,COND=(4,LT)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN DD *
    DSN SYSTEM(ABCD)
    RUN PROGRAM(DSNTIAD) PLAN(DSNTIA81) -
        LIB('DSN810.RUNLIB.LOAD')
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSIN DD *
```

and (intervening lines have been deleted)

---

```
//PH01S13 EXEC DSNUPROC,PARM='ABCD,DSNTEX',COND=(4,LT)
```

---

Example 4-6 shows part of the sample job DSNTEJ1 which is found in your customized SDSNSAMP data set output from the invocation of the installation CLIST. It has a common invocation of the DSN command processor in which the SYSTEM parameter is ABCD. While it will no longer be your subsystem name, it is your group attach name, and can be submitted on any z/OS image that has a member of that group active.

The other example is an invocation of a DB2 utility. The subsystem to which you were attaching in non data-sharing is now the group attach name and it also can be submitted on any z/OS image that has an active member of your group.

Specification of a group attach name on the JCL is preferred, as you have maximum flexibility about where the job runs without the need to change the JCL if you want to run it on a different z/OS image. You also have a single copy of the JCL versus a copy for each member. This may not seem like a problem with 2-way data sharing, but it certainly is when you have a 6-way group!

## 4.5.1 Additional tasks

Here we describe the additional tasks that you have to perform:

1. Make sure the CFRM policy that contains definitions for the DB2 structures has been activated. This can occur long before you enable data sharing. The structure sizing and definition are described in 3.1, “Structure sizing” on page 40 and 3.5, “CFRM policy” on page 60.
2. Make sure that the renamed DB2’s member name (DBP1) has been incorporated in the automation procedures active in your environment, such as the ARM policy.
3. You can now start DBP1 in data sharing mode (one way). DB2 will perform a group restart. It will build a lock structure and SCA during the restart (a GBP is not allocated until the second member is started and GBP-dependency occurs for the first time). After DB2 is started, you can issue the following commands to display the DB2 group information:

```
D XCF,STR,STRNM=DSNDB2P*  
-DBP1 DIS GROUP
```

### ***DSNTIJTM***

In a data sharing environment you do not use the temporary work file database called *DSNDB07* any longer; you use a new temporary work file database according to your own naming convention. See 4.1, “Naming conventions” on page 66” for more information about naming convention. With the DSNTIJTM JOB, you create a new temporary work file database. It drops DSNDB07 and creates WRKDBP1 in DBP1.

## 4.6 Adding the second member

When you finish the installation panels described in “Adding new members” of Chapter 3. “Installing and enabling DB2 data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, you have the following JOBS in your *hlq*.DBP2.SDSNSAMP data set:

```
DSNTIJDE DSNTIJFT DSNTIJGF DSNTIJID DSNTIJIN DSNTIJMV DSNTIJTM  
DSNTIJUZ
```

There are now considerably fewer PDS members than there were when you enabled data sharing with the original data sharing member.

Here are the remaining steps to follow:

1. Add the same information to RACF and to the IEFSSNxx PDS member of SYS1.PARMLIB, though you should have already done this at the same time you added the information for the originating data sharing member.
2. When you complete DSNTIJIN, DSNTIJID and DSNTIJUZ successfully, you can start the newly added DB2 sharing member on another z/OS image in the Parallel Sysplex. You can see your added member by issuing the command:  
**cmdprefix DIS GROUP**
3. Run DSNSTIJTM to create the new temporary work file database for the new member, DBP2.

## 4.7 Removing a member

If you want to remove a member from your data sharing group, either temporarily or permanently, stop that member using MODE(QUIESCE). The member will be “dormant” until you restart it. A member that is quiesced cleanly can remain dormant forever. A quiesced member will still appear in displays and reports. Bring the quiesced member’s log and BSDS data sets to your disaster recovery site.

The BSDS data sets and logs for the quiesced member must be kept until such time as they are no longer needed for recovery. The BSDS is also needed for group restart.

If you plan to keep this member dormant forever, you can delete the BSDS data sets when they are no longer needed for recovery. Group restart will then issue the following message:

**DSNR020I csect-name START MEMBER member, OR REPLY 'NO' OR 'QUIESCED'**

When you respond with QUIESCED, DB2 issues the following message:

**DSNR030I csect-name WILL CONTINUE WITHOUT THE member MEMBER'S LOG,  
REPLY 'YES' OR 'NO'**

You respond **YES**.

For more detailed information about removing a member from the group, see “Removing members from the data sharing group” in Chapter 3. “Installing and enabling DB2 data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.





## Dynamic workload balancing

Dynamic workload balancing is an important factor in achieving business benefits from data sharing. Continuous availability, flexible configuration, scalable growth and managing workload spikes provide most advantages when workload can be routed dynamically to the images in the Parallel Sysplex that have the most capacity or best throughput.

This chapter describes the components, processes, and features that allow you to achieve dynamic workload balancing in a DB2 data sharing environment.

We discuss the following topics:

- ▶ Objectives of workload balancing
- ▶ Workload Manager
- ▶ Dynamic virtual IP addressing (DVIPA)
- ▶ Sysplex Distributor
- ▶ Distributed data facility (DDF)
- ▶ Stored procedures
- ▶ Batch work
- ▶ WebSphere
- ▶ CICSplex Systems Manager (CP/SM)
- ▶ IMS Transaction Manager (IMS TM)

## 5.1 Objectives of workload balancing

There are several objectives for using dynamic workload balancing:

- Availability** The objective is to route incoming work to resources that are currently active, and to avoid resources that are experiencing planned or unplanned outages.
- Capacity** The objective is to route incoming work to resources that provide the best response time or transaction throughput, and to avoid resources that are overcommitted
- Workload spikes** The objective is to allow work to flow to resources added to handle specific workload spikes without re-coding applications or changing JCL.

Workload routing begins in the network layer. See Figure 5-1.

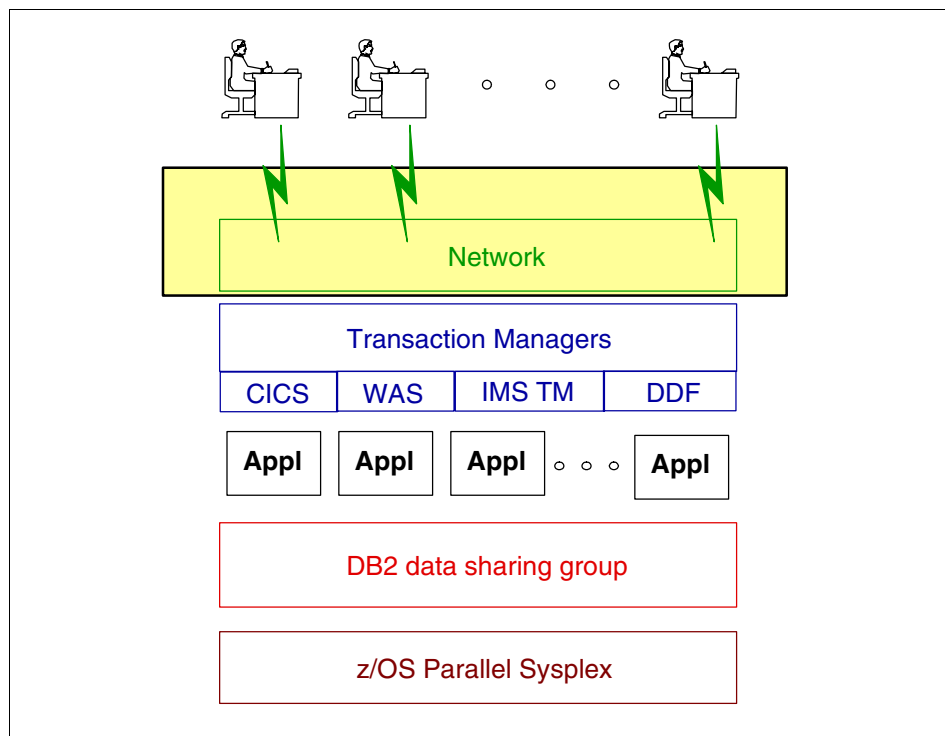


Figure 5-1 Workload routing in the network layer

Requests received from TCP/IP or from VTAM can be routed among resources in the Parallel Sysplex. TCP/IP traffic can be directed to the target resource using dynamic virtual IP addressing (DVIPA). DVIPA and Sysplex Distributor, a component of IBM Communication Server, work with z/OS Workload Manager (WLM) to distribute incoming network requests based on current system conditions. VTAM traffic can be directed using VTAM generic resources.

Once the network layer has routed the incoming requests, additional balancing may occur, depending on whether the requests are for transaction manager services or directly to DB2 via the distributed data facility (DDF). Some of the transaction managers can also route requests or balance workload among their resources. DDF requests are only balanced in the network layer.

The various transaction managers that you use can each play a role in accomplishing workload balancing. How you define them also determines to what extent workload balancing can be achieved dynamically.

Figure 5-2 shows the abstract view of the transaction manager layer. In a way that is similar to the Parallel Sysplex and data sharing group layers, the transaction manager layer can have instances in any image in the Parallel Sysplex, so transactions can be executed on any of these instances.

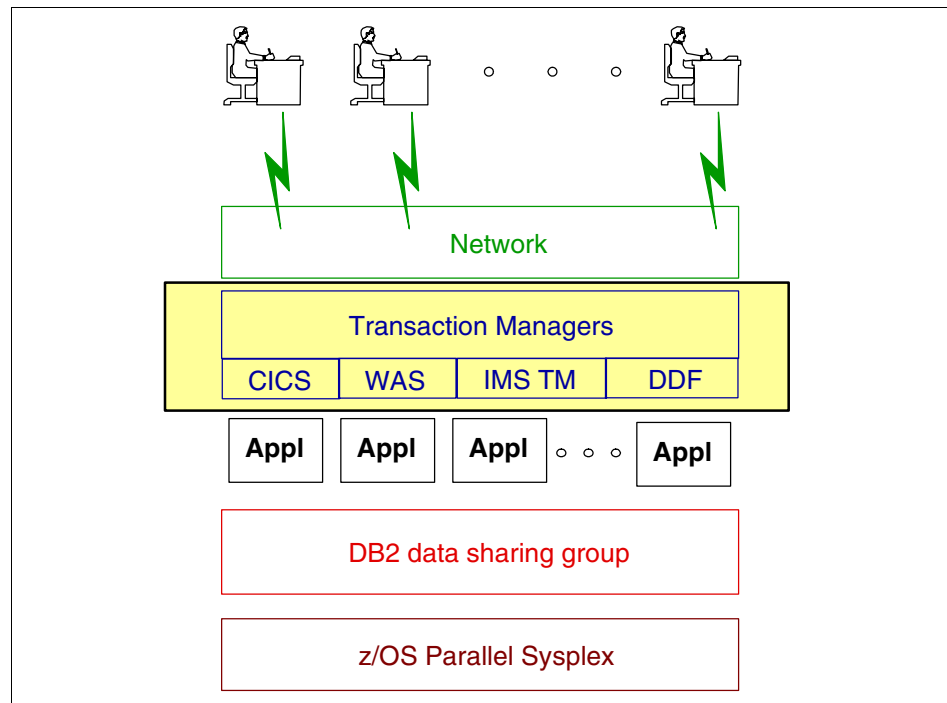


Figure 5-2 Workload routing in the transaction manager layer

CICS<sup>tm</sup> and CICSplex® Systems Manager, IMS<sup>tm</sup> Transaction Manager, using IMS Connect routing or shared message queues, WebSphere Application Server, and WebSphere MQSeries®, all participate in what we call the transaction manager layer. Each of these has facilities to route work to available resources. Some of these facilities are independent of network interaction; other facilities rely on network interaction to route work.

For example, MQSeries receives messages after network routing, but it can place those messages on shared queues. Doing this allows applications to pull from these queues from any image in the Parallel Sysplex. Although DB2's distributed data facility (DDF) can also be considered to be in this layer, it is not a transaction manager and it relies on the network layer for workload routing.

Each of the transaction managers can accept and execute work on any image in the Parallel Sysplex where they are defined. From a DB2 perspective, each transaction manager instance will attach to the DB2 member that is coresident on that z/OS image. If you have more than one transaction manager in one z/OS image, both can attach to the same DB2 member on that image. While it is likely that your environment will have more than one such source of DB2 workload, we limit the discussion in the sections below to one source of work at a time.

## 5.2 Workload Manager

The z/OS Workload Manager (WLM) plays a key role in achieving dynamic workload balancing benefits in a data sharing environment. This is in addition to WLM's role of managing performance goals within a z/OS image. WLM directs system resources to achieving business goals as described in a WLM policy. There is a single active policy for a sysplex, and WLM uses that policy to direct resources to achieving the business goals.

By interacting with the various subsystems, including DB2, WLM provides information about system capacity and availability of resources in the Parallel Sysplex. Generally the z/OS systems administration or performance staff will manage the WLM policy. Refer to *z/OS MVS Planning: Workload Management*, SA22-7602 for more information. The redbook, *System Programmer's Guide to: Workload Manager*, SG24-6472, is also very helpful.

The following Web site is the home page for WLM:

<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>

From a data sharing perspective, WLM affects the following resources:

► DB2 Distributed Data Facility (DDF):

WLM manages DDF threads against their performance goals, but also provides information to indicate to which DB2 member a new DDF connection request should be directed. The key to availability for DDF traffic is Dynamic Virtual I/P Addressing, or DVIPA, coupled with the Sysplex Distributor.

► WebSphere resources:

WebSphere Application Server can access DB2 data locally from System z or remotely from System z or other platforms. In the case of local access, WLM is involved in balancing network requests between WebSphere Application Server instances on different images in the Parallel Sysplex.

Remote WebSphere Application Server instances access DB2 through DDF and receive the same balancing benefits that DDF work does.

► CICS Transaction Server:

WLM provides information to Sysplex Distributor, which requesters can use to balance TCP/IP traffic among CICS regions.

WLM provides information to VTAM Generic Resources to balance logons in an SNA environment. WLM can manage resources (CPU, I/O, storage) for CICS.

► CICSplex Systems Manager (CP/SM):

The CP/SM element of CICS/TS is a system management tool to manage multiple CICS systems as if they were a single system. CP/SM recommends to the CICS TOR which AOR to route the transaction to in an MRO environment.

CP/SM Real-Time Analysis (RTA) function provides automatic, external, notification of conditions in which you have expressed an interest.

► IMS Transaction Manager:

WLM provides information to Sysplex Distributor to allow clients to direct TCP/IP requests for IMS services to the appropriate IMS Connect, which is the IMS interface to TCP/IP.

In an SNA network, WLM provides information to VTAM Generic Resources to balance logon requests between IMS subsystems. WLM also interacts with the IMS MPRs and BMPs to help get work through the system.

► Batch work:

WLM monitors batch performance goals and can start initiators throughout the Parallel Sysplex to allow batch jobs to execute. By starting initiators on more than one z/OS image, WLM can cause batch work to be executed on any image where the appropriate initiators are available.

## 5.3 Dynamic virtual IP addressing (DVIPA)

For data sharing groups in a TCP/IP network, DVIPA is the recommended approach to meeting the objectives of dynamic workload balancing. The goal is to make the members of the data sharing group easily accessible for requests from the network while making minimal changes to the network clients.

### Network requesters do not have to change

In a non-data sharing environment, network requesters will specify a location name for database access and either an IP address or a domain name, preferably both, for the network access to the server where the database resides. In a data sharing environment, the location name should be the same as the location name of the originating DB2 member, but it now refers to more than one member. Keeping the original location name avoids changing the location name in the configuration settings in all the distributed requesters.

To allow distributed requesters to connect to any member of the data sharing group, we must allow the IP address that they already use to refer to any member of the group. We can do this by defining the IP address of the original DB2 subsystem as the virtual IP address for the group, and assigning unique IP addresses for each member. Again, this means the network requesters do not have to change their configurations.

### DB2 definitions

Because you will use the original location name to refer to the data sharing group and the original IP address to refer to the group, you must uniquely identify each DB2 to TCP/IP and still allow network requesters to reach a specific member in case it becomes necessary to handle communication failures during two-phase commit processing.

Besides specifying the listening port and resync ports in the members' bootstrap data sets, as described in 4.1.6, "Distributed Data Facility (DDF) related parameters" on page 69, you will have to bind each of the DB2 members to the listening port and resync port in the TCP/IP *PORT* statements. Make sure you specify *SHAREPORT* for each member on the common listening port.

In addition, specify the *VIPADYNAMIC* definitions with *VIPADEFINE RANGE* for each specific IP address. Also, in the *VIPADYNAMIC* definition, make sure the *VIPADEFINE* and *VIPADISTRIBUTE DEFINE* for one member and *VIPABACKUP* for each additional member specify the group dynamic VIPA.

See *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417, Chapter 4, section *Dynamic VIPA network addressing* for a diagram and further references.

## 5.4 Sysplex Distributor

Sysplex Distributor is a component of IBM Communication Server that provides *routing information* to clients requesting access to resources in a Parallel Sysplex. WLM sysplex routing services provide routing information, including preferred resources, on available servers. TCP/IP domain name server (DNS) and dynamic virtual IP addressing (DVIPA) use Sysplex Distributor to receive information about the servers in a Parallel Sysplex.

When Sysplex Distributor requests routing information from WLM, it finds out which servers are available and the relative weights of the servers. Sysplex Distributor then provides this information to the clients. The client then uses this information to balance connection requests among available servers.

Sysplex Distributor provides routing information to client requesters using TCP/IP to access enterprise servers including:

- ▶ DB2 DDF using the DB2 Universal JDBC™ Driver and Type 4 connections
- ▶ DB2 DDF using DB2 Connect™ V8.2
- ▶ CICS
- ▶ IMS

For more information on Sysplex Distributor, see *TCP/IP in a Sysplex*, SG24-5235 and *Leveraging z/OS TCP/IP Dynamic VIPAs and Sysplex Distributor for higher availability*, GM13-0165.

## 5.5 Distributed data facility (DDF)

Clients can access DB2 via DDF using TCP/IP or SNA. TCP/IP is the strategic network alternative and is the focus of our discussion here. If you have an SNA network that handles DDF requests, see the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 for details on using either member routing (preferred) or VTAM generic resources to achieve workload balancing.

Clients in a TCP/IP network use distributed relational database architecture (DRDA) requests to access DB2. The client can either access DB2 directly or via a gateway. In either case the client may take advantage of sysplex routing services, as described above in 5.4, “Sysplex Distributor” on page 95. Support for sysplex routing services is provided in the DB2 Universal JDBC Driver and in DB2 Connect 8.2. If you are not using either of these, then the client you are using must have its own support of sysplex routing services to receive maximum benefit from workload balancing. The following scenario assumes that you are using sysplex routing services such as those provided by DB2 Connect.

A client will use its configuration files, or those on a gateway, to identify the DB2 data sharing group with a location name and an IP address or DNS name. As described in 5.3, “Dynamic virtual IP addressing (DVIPA)” on page 94, these should remain the same after implementing data sharing.

When the client issues a connection request, for example, **CONNECT TO DB2PLOC**, the local or gateway configuration will direct the request to the virtual IP address of the data sharing group. Sysplex Distributor will provide weighted values for the systems in the Parallel Sysplex that it received from WLM. These weighted values and the specific IP addresses of the data sharing group members are returned to the client in response to the connection request. The client then chooses which member to specify.

It is important to access Sysplex Distributor on the first connection, because that is the fastest way to identify which members of the data sharing group are available at the time the request is processed. Subsequent traffic between the client or gateway and the data sharing group can be balanced between members of the data sharing group without referencing Sysplex Distributor.

Figure 5-3 shows both an individual client and clients connecting through a gateway. In both cases the first access is through Sysplex Distributor to identify which members are currently available.

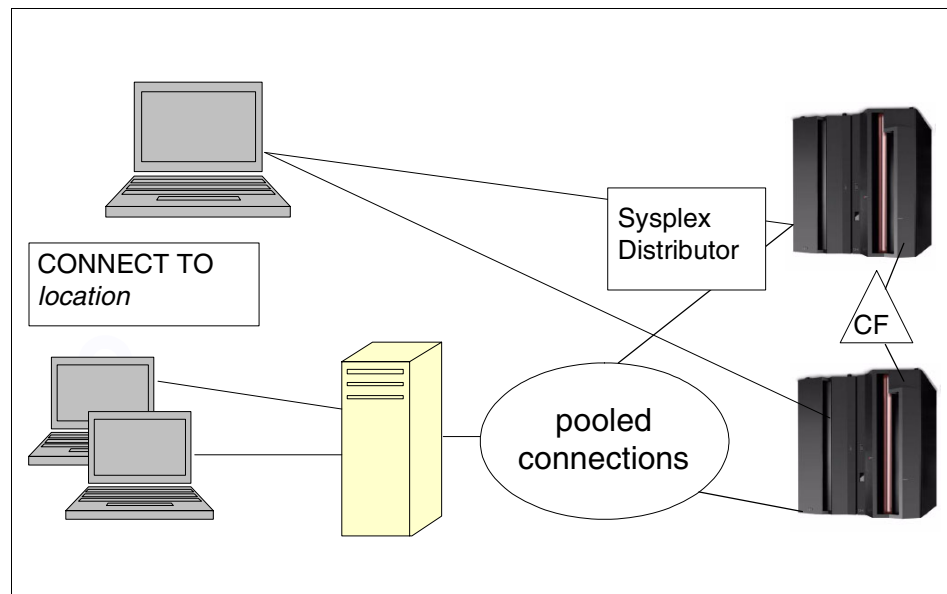


Figure 5-3 Clients balancing DDF requests into a DB2 data sharing group



In the case of the gateway, it is possible for the gateway to maintain pooled connections with the members of the data sharing group. This can reduce overhead and response time by reducing the time and processing required to establish connections on a repeated basis. This savings applies at both the gateway and the members of the data sharing group.

## 5.6 Stored procedures

Stored procedures themselves do not change with DB2 data sharing. What changes is that the applications that issue the *CALL* statements may be running anywhere in the Parallel Sysplex. This has several implications. One implication is that to achieve the desired level of availability and to support workload balancing, each member of the data sharing group must be able to start any stored procedure. The other implication is that the application considerations for data sharing apply to stored procedures just as they do to any other application.

Stored procedures that execute in WLM-established stored procedure address spaces are assigned to an application environment. The definition of the application environment includes a parameter that specifies the DB2 subsystem name. We recommend that you share SYS1.PROCLIB among the images in your Parallel Sysplex. This will allow you to maintain one copy of the JCL procedure for the application environment.

If you share SYS1.PROCLIB between z/OS images, or if you want a common JCL procedure for an application environment that may be invoked by any DB2 member, then the JCL procedure for the application environment must specify the subsystem parameter to allow any of the DB2 member names to be substituted. This parameter is DB2SSN, which appears in the PROC statement in the JCL that defines the application environment. Specifying DB2SSN=&IWMSSNM will allow that JCL procedure to associate with whatever member invokes the stored procedure.

You should also specify DB2SSN=&IWMSSNM in the *Start Parameters* field on the *Create an Application Environment* WLM panel that defines the JCL for an application environment.

If you do not share SYS1.PROCLIB between your z/OS images, and only one DB2 member will use the JCL procedure to define an application environment, then you do not need to code the symbolic, &IWMSSNM.

Starting in DB2 for z/OS V8, you cannot create stored procedures that execute in a DB2-established stored procedure address space. You can only invoke these stored procedures if they were created in DB2 V7 or before. We recommend that you convert these stored procedures to run in WLM-established stored

procedure address spaces. Until you complete that conversion, for stored procedures that execute in the DB2-established stored procedure address space, all that is required is that this address space be started by each member of the data sharing group.

Refer to *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083 for more information on DB2 stored procedures.

## 5.7 Batch work

Batch workloads, whether traditional batch or IMS batch message programs (BMPs), can take advantage of Parallel Sysplex resources if they can be started on more than one z/OS image. In 2.5, “Application considerations in data sharing” on page 35 we discussed reasons why you might modify your batch processes to take advantage of the benefits of the data sharing architecture.

There are two approaches to balancing batch work across the images of a Parallel Sysplex. One approach uses WLM-managed JES initiators (MODE=WLM), for JES2 and JES3. The other approach relies on scheduling software that is customized for your environment.

Typically, a batch job can execute on any z/OS image that has available initiators for the job's class. WLM monitors the batch jobs against their performance goals. If batch jobs are not meeting their goals, WLM can start more initiators for the job classes defined with WLM-managed initiators (MODE=WLM). As long as the initiators are defined on each z/OS image, the batch jobs can be started on each image.

Scheduling programs allow for fairly detailed definition of the batch job stream including predecessor and successor jobs and interdependencies. Your scheduling program should be aware of the Parallel Sysplex and its resources to take the most advantage of your available capacity. The IBM Tivoli® Workload Scheduler provides this function to support execution of batch jobs across the resources of the Parallel Sysplex.

Tivoli Workload Scheduler for z/OS (TWS) has an interface with WLM that supports the critical path concept of job scheduling. For example, if TWS detects that a critical job is running late, TWS calls a documented WLM interface to move the job to a higher-performance service class. The job receives more system resources and completes in a quicker time. The delay in the batch workload is therefore reduced.

## 5.8 WebSphere

WebSphere Application Server can access DB2 locally, from the same System z environment, or remotely, either from System z or other platforms. In the case of a local WebSphere Application Server instance connecting to a local DB2 with Type 2 connectivity, there is no workload balancing between WebSphere Application Server and DB2.

For remote WebSphere Application Server instances, access to DB2 via DDF can take advantage of sysplex routing services in Sysplex Distributor and the flexibility of DVIPA to establish connections. WebSphere Application Server uses connection pooling and connection concentration to balance workload dynamically. This function is available in the DB2 Universal JDBC Driver Type 4 connectivity.

When a WebSphere transaction issues an SQL request it gets assigned a connection in the database connection pool. The DB2 Universal JDBC Driver Type 4 connectivity provides connection concentration function for the connections in the pool. Each DB connection in the pool is assigned to a logical connection (LC1, LC2, or LC3 in Figure 5-4). It can also *concentrate* the logical connections among fewer transports. The logical connections are eligible to be assigned to different transports after a commit.

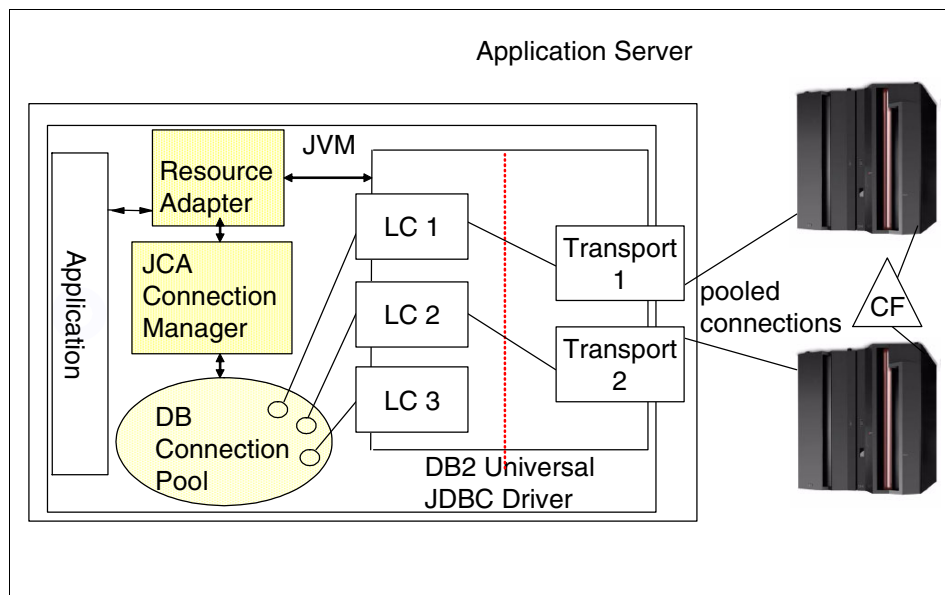


Figure 5-4 Remote WebSphere Application Server using DB2 JDBC Universal Driver to access a data sharing group

The transports use pooled connections to the members of the data sharing group. WebSphere Application Server can balance the connection requests across the pooled connections to the data sharing group based on availability and performance feedback. This balancing occurs dynamically and is transparent to the application processes.

For more information on WebSphere Application Server and connection concentration and use of Type 4 connectivity in the DB2 Universal JDBC Driver, see the *DB2 for z/OS Application Programming Guide and Reference for Java™*, SC18-7414.

## 5.9 CICSplex Systems Manager (CP/SM)

CICS and CP/SM take advantage of workload balancing performed in the network layer by either TCP/IP or VTAM. TCP/IP and the Sysplex Distributor can balance incoming CICS requests between CICS terminal owning regions (TORs). For SNA network traffic, VTAM uses generic resource definitions to balance logons between CICS TORs.

Within the CP/SM environment transaction requests that arrive in one TOR can be routed to an application owning region (AOR) in the same z/OS image or in another z/OS image. This routing is based on CPSM information about CICS resources in target regions and, if running in goal mode, about how well target regions are meeting their goals.

Figure 5-5 illustrates both balancing in the network layer and workload routing from the TORs to AORs across the sysplex.

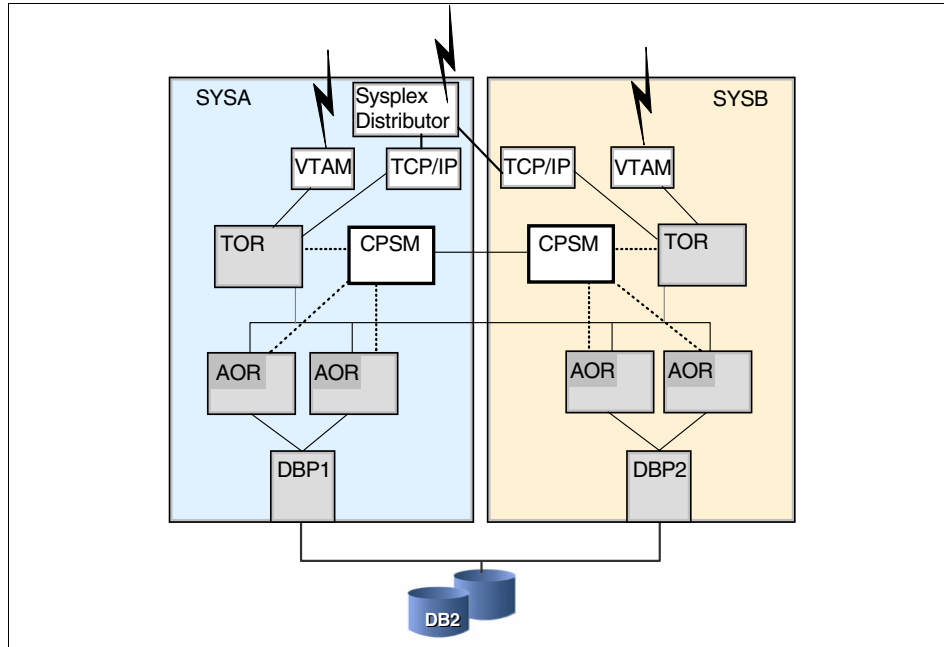


Figure 5-5 CICS and CP/SM in a Parallel Sysplex environment

## 5.10 IMS Transaction Manager (IMS TM)

IMS takes advantage of workload balancing performed in the network layer by either TCP/IP or VTAM. TCP/IP and the Sysplex Distributor can balance incoming IMS messages between IMS Connect regions. An IMS Connect can send a message to an IMS based on any algorithm that the user implements in an exit routine.

Typically, IMS Connect will send messages to one IMS subsystem, but can route messages to other subsystems based on subsystem availability. For SNA network traffic, VTAM uses generic resource definitions to balance logons between IMS subsystems.

Figure 5-6 shows network requests distributed across two IMS subsystems in a Parallel Sysplex. IMS transactions can exploit all the advantages of DB2 data sharing as well as IMS DB data sharing.

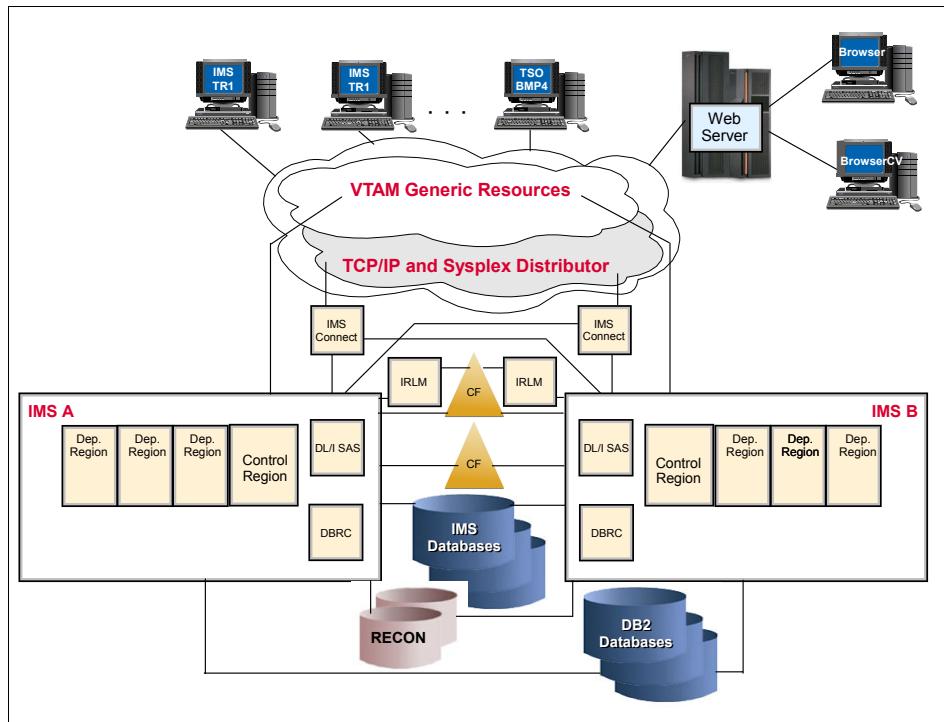


Figure 5-6 IMS Subsystems in a Parallel Sysplex

IMS can further distribute requests to handle overflow conditions. If a message that reaches an IMS subsystem cannot be scheduled right away, the message can be placed on a shared message queue in the coupling facility, allowing another IMS subsystem to schedule the message. IMS shared queues can thus handle intermittent peaks dynamically.

## 5.11 Dynamic workload balancing best practices

- ▶ Use Dynamic virtual IP addressing (DVIPA) and Sysplex Distributor for availability and failover. This combination will ensure that network connection requests are routed to an available member and, if there is more than one member available, to the one with the greater capacity.
- ▶ Share SYS1.PROCLIB among the members of the data sharing group and use one copy of the JCL for each stored procedure.
- ▶ Ensure that all your DB2 stored procedures run in WLM-established stored procedure address spaces.
- ▶ For DB2 for z/OS requesters to a data sharing group, use TCP/IP connectivity and take advantage of the benefits of DVIPA.
- ▶ If you still use an SNA network, use member routing for requests from a DB2 for z/OS requester to a data sharing group. Member routing provides similar sysplex routing services to DVIPA and Sysplex Distributor. Avoid using VTAM generic resources.







# Operations

DB2 data sharing introduces several new and/or changed concepts to DB2 operations. We strongly recommend that you become familiar with these areas of operational impact. You should modify your existing operational procedures to incorporate the necessary changes.

We discuss the following topics:

- ▶ Recovery of DB2 objects
- ▶ Component failure
- ▶ Sysplex failure management (SFM) policy
- ▶ Automatic restart manager (ARM)
- ▶ Restart Light
- ▶ Commands
- ▶ Multi-system DB2 diagnostic dumps
- ▶ Disaster recovery
- ▶ Rolling maintenance

## 6.1 Recovery of DB2 objects

The RECOVER utility works basically the same way for data sharing as it does for the non-data sharing environment. The most recent image copy is restored and the updates that have been logged since the image copy was created are applied. The major difference is that the log records that need to be applied could belong to several members of the data sharing group. Also, the DB2 subsystem that is performing the recovery needs to ensure that the log records are applied in the proper sequence. The RECOVER utility can be run on any active member of the DB2 data sharing group.

In this section we define the following terms:

- ▶ Log record sequence number (LSRN)
- ▶ Group buffer pool recovery pending (GRECP)
- ▶ Logical page list (LPL)

Best practices for GRECP/LPL recovery are also provided.

### 6.1.1 Log record sequence number (LSRN)

LSRN is a value derived from the stored clock timestamp and synchronized across the members of a data sharing group by the Sysplex Timer. This is the mechanism used in data sharing to provide the ordering of log records that have been updated by any member in the group. In non-data sharing, relative byte address (RBA) is used for ordering of log records and recovery.

### 6.1.2 Group buffer pool recovery pending (GRECP)

If a coupling facility fails, or if all members have lost connectivity to a GBP, data sets with pages in the affected GBP are placed by DB2 in GRECP exception status. Their recovery consists of applying changes that are recorded in the log to the page set.

DB2 automatically recovers GRECP page sets when the group buffer pool is defined with AUTOREC (YES). The exception to this is *when DB2 detects GRECP conditions during restart in DB2 V8*.

With duplexed GBPs, the only time you should experience GRECP is when both coupling facilities (CF) are unavailable. If a single CF fails, fallback to simplex occurs within a few seconds and no GRECP occurs.

### 6.1.3 Logical page list (LPL)

The LPL exception condition is set if pages cannot be read from or written to the group buffer pool. In non-data sharing, pages can only be set LPL when they cannot be read from or written to disk because of a transient disk problem. Messages such as DSNB250E, DSNB303E, DSNB311I, and DSNB312I indicate that pages have been written to the LPL and provide information as to why they were put in the LPL. DB2 also automatically recovers LPL conditions *unless it is detected during restart in DB2 V8*.

### 6.1.4 Recommendations for GRECP/LPL recovery

The most common reasons that you may need to manually recover GRECP/LPL in DB2 V8 occur under the following circumstances:

- ▶ When you perform offsite disaster recovery testing (see *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370)
- ▶ For a Parallel Sysplex wide outage that could occur with a power failure when auxiliary power is not available

#### Notes:

- ▶ If all the z/OS images were to fail, there is normally no GRECP/LPL recovery needed on restart as long as the CFs remain available.
- ▶ DB2 Version 9.1 for z/OS performs GRECP/LPL recoveries even when the conditions are detected during restart.

To reduce this unplanned outage, you need to consider these settings:

- ▶ Frequency of group buffer pool checkpoints
- ▶ Group buffer pool castout threshold
- ▶ Group buffer pool class castout threshold
- ▶ Use of AUTOREC(YES)

If GRECP/LPL conditions are set during DB2 restart, you must manually remove them. You normally do this by issuing START DATABASE commands, as it is much faster than alternate methods such as RECOVER utility or LOAD utility. The command reads the active logs and applies all records that had not been externalized to disk.

Best performance is obtained by the use of *Fast Log Apply (FLA)*, a process enabled by the DSNZPARM LOGAPSTG in the DSN6SYSP macro. This can be specified on the Log Apply Storage option of the DSNTIPL install panel. The maximum value that can be specified is 100, which refers to megabytes of storage in the DBM1 address space that DB2 can use for FLA invocations.

**Note:** Some users have hesitated to implement FLA because they fear running out of virtual storage in the DBM1 address space.

If the storage size specified is not available when FLA attempts to allocate it, DB2 tries to allocate a smaller amount. If there is insufficient storage for FLA, DB2 reverts to standard log apply.

Each START command acquires a 10 MB log buffer and sorts the list of objects in the START command in database/table space/page/LRSN sequence. One subtask is created for each of the objects in the list, up to 98 (More than 98 objects will wait until one of the subtasks are available).

Not only are the log records ready to be applied in sequence, but also, the pages of the data set are acquired through a process called list prefetch. This capability allows a recovery that is up to 10 times faster than without FLA. Without FLA, each log record is read sequentially, the applicable page is read synchronously into the buffer pool and applied. The next log record, which is likely for a different table space, is read, and the process continues.

Create your START commands so that you process multiple objects per command. Issue no more than 10 START DB commands per DB2 member because the subsequent commands will not use FLA. We recommend the commands shown in Example 6-1, which will resolve both GRECP and LPL conditions at the same time.

---

*Example 6-1 Issuing START commands*

---

```
-START DB(DSNDB01) SP(*)  
-START DB(DSNDB06) SP(*)  
-START DATABASE(db1) SPACE(*)  
-START DATABASE(db2) SPACE(abc*)
```

---

You should always start the DB2 Directory (DSNDB01) and DB2 Catalog (DSNDB06) first, as shown in Example 6-1. Following their successful restart, issue the commands for other databases in your own priority sequence.

If the number of objects in a database is fewer than 90 or so, you can issue the database form of the command. If you have several hundred objects in a database, then you should split them as shown in the last form of the command, making sure that you do not exceed the limit of 98. In fact, it is better to restrict it to about 50-70 objects, which allows for future growth without your having to recalculate the number each time.

**Note:** Although you predefine the list, nevertheless, at the time of the disaster the number of objects actually in GRECP/LPL will vary. In a list of 50-70 objects, there may be none, or 5, but that is OK.

You should have Install SYSADM authority for the commands. If you do not, DB2 will check for your authorization, and if the directory objects (DSNDB01) are in GRECP, it is likely to fail. Because Install SYSADM is defined in DSNZPARM, it is validated by that means.

If you fail to recover GRECP pages by START DATABASE command, you can recover the pages by using the RECOVER utility. See Chapter 23. “RECOVER” of *DB2 for z/OS Version 8 Utility Guide and Reference*, SC18-7427. It may take a long time to recover objects in GRECP status, depending on the workload on your system.

### 6.1.5 Best practices for GRECP/LPL recovery

Here are some best practices recommendations for GRECP/LPL recovery:

- ▶ Use DSNZPARM LOGAPSTG=100 to enable fast log apply recovery code.
- ▶ Place active logs on separate devices (for availability) to achieve the best performance for recovery since both copies of the log are read.
- ▶ Submit 10 jobs per member (each job processes 10 MB of the log).
- ▶ Issue **-STA DB(DSNDB01) SP(\*)**
- ▶ Issue **-STA DB(DSNDB06) SP(\*)**
- ▶ Start other databases in priority sequence with pre-coded start commands with fewer than 70 objects in the list (to allow for growth within the database). If a database has more than that, then pre-code the start database command with wildcard, that is, SPACE(ABC\*) to limit the list.

Do not issue start commands for each individual object, as that will result in poor disk performance due to massive contention on the DB2 active log.

## 6.2 Component failure

One major reason to implement Parallel Sysplex and DB2 data sharing is to provide continuous availability during both scheduled and unscheduled outages to any component. It is necessary to set up the environment so that a component failure has minimal or no visible impact to the availability of the applications to the users. High availability can be facilitated by properly defining the policies that control the actions taken by the system when a failure occurs. These include the CFRM policy, SFM policy, and ARM policy.

For more information about the policies and the ways to define them, see *z/OS V1R7.0 MVS Setting Up a Sysplex*, SA22-7625.

The following section provides information for the following circumstances:

- ▶ DB2 subsystem failure
- ▶ z/OS system failure
- ▶ CF failure

## 6.2.1 DB2 subsystem failure

The locks held by a failing DB2 member of a data sharing group environment that fails become retained locks. These retained locks are necessary to protect data in the process of being updated from being accessed by another active member of the group. Retained locks should be released as soon as possible to allow access to all data by the surviving members of the group.

### Normal restart

For this, you simply restart DB2 using automation, either your own or automatic restart manager (ARM). Automation is much faster than manual operation. Time to restart becomes a critical path in data sharing, because the DB2 members that are active can experience timeouts for workload due to retained locks held on a failed member. Normal restart for a member of data sharing group is exactly the same as for a non-data sharing group. For more information about normal restart, see “Normal restart for a data sharing member” in Chapter 5. “Operating with data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

### Group restart

A group restart is initiated by DB2 if either the SCA or the lock structure (or both) are lost (CF failure or deactivation), and can not be rebuilt on the other coupling facility. When this happens, all members of the data sharing group terminate abnormally. Group restart rebuilds the information from each member's logs. Any member that is restarted will detect the situation and will rebuild the SCA and lock structures, but it is best to start all members at once. Before performing a group restart, you need to delete a persistent structure or a failed-persistent connection.

For more information refer to “Group restart” in Chapter 5. “Operating with data sharing” in *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

## 6.2.2 z/OS system failure

If the underlying z/OS system fails, the DB2 subsystem in the z/OS image should be restarted on another z/OS image in the Parallel Sysplex as soon as possible so that retained locks are released. You can automate this cross-system restart if your z/OS systems team has implemented automatic restart manager (ARM) policy.

In non-data sharing, some users wait for an IPL of the failed z/OS image. For data sharing users, this is too long to wait. The focus needs to be changed from the failed image (and its DB2) to the DB2 members that are active on another image. The active workload can incur timeouts and failures due to the retained locks still held by the member that remains down.

Once the DB2 subsystem has restarted in another z/OS image, all of the retained locks will be released.

Refer to *z/OS MVS Setting Up a Sysplex*, SA22-7625 for information about how your z/OS systems staff can create an ARM policy.

## 6.2.3 CF failure

The values you specified for your CF structures play a large part in determining the recovery actions that are taken when a CF fails. This refers to a single failure and not to a double failure. Refer to 3.3, “Duplexing” on page 53 and 3.4, “CF configuration alternatives” on page 56.

From our CFRM policy example shown in 3.5, “CFRM policy” on page 60, we have two CFs in the preference list. Duplexing is enabled, and a loss of 1% of the weight, as defined in the SFM policy, results in a rebuild of the structure to the alternate CF in the preference list (PREF).

For duplexed GBPs, loss of CF will fall back to simplex within seconds and the outage may not even be noticed.

For the Lock and SCA structures that are not duplexed, rebuild occurs automatically (normally within a minute or two depending on the amount of activity at the time of the failure) to the second CF in the preference list.

See *z/OS MVS Setting Up a Sysplex*, SA22-7625 for information about how to create CFRM and SFM policies. See “Recovery from coupling facility failures” in Chapter 5. “Operating with data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 for more information about CF failures.

## 6.3 Sysplex failure management (SFM) policy

If you have not implemented the sysplex failure management (SMF) policy, you will realize its benefits when a z/OS image on which a DB2 member in the group runs appears to hang. A member of the sysplex that has not updated its status within the specified failure detection interval needs to be partitioned out promptly.

It is rarely possible to save a failed system through operator action after XCF has detected failure of the system. Automatic removal of the failed system from the sysplex is preferable to avoid “sympathy sickness” from the lack of access to resources locked by the failing system. The objective is to minimize timeouts from surviving DB2 members. When your z/OS system staff implements the SMF policy, it will accomplish this goal. We recommend the following values for your z/OS systems staff:

- ▶ Implement ISOLATETIME instead of PROMPT.
- ▶ Assign appropriate weights to the systems according to their importance.
- ▶ Specify an ISOLATETIME of zero in the SMF policy.

## 6.4 Automatic restart manager (ARM)

Automatic Restart Manager (ARM) provides a quick restart capability without any operator action. Locks held by the failed members, called retained locks, will be released as soon as the member is restarted. Releasing the retained locks quickly is very important to provide high data availability to applications running on other members while maintaining data integrity. ARM can rebuild CICS regions associated with the DB2 to resolve in-doubt units of work.

Your z/OS systems staff will set up the ARM policy if ARM has been implemented in your environment. See *z/OS MVS Setting Up a Sysplex*, SA22-7625 for information about how to create ARM policy.

## 6.5 Restart Light

If the failed DB2 subsystem was running on a z/OS image which is no longer available, it is critical to restart the failed DB2 in another z/OS image in the same Parallel Sysplex (where another member might be active) in order to release the retained locks. Another z/OS image may not have the resources to handle the workload of an additional DB2 subsystem. RESTART(LIGHT) enables DB2 to restart with a minimal storage footprint to quickly release retained locks and then terminate normally.



The ARM policy described previously also has the capability to perform an automated restart using the light option. You specify LIGHT(YES) within the RESTART\_METHOD(SYSTEM) keyword for the DB2 element name. For example:

```
RESTART_METHOD(SYSTEM,STC,'cmdprfx STA DB2,LIGHT(YES)')
```

However, Restart Light does not release the locks that are held by postponed abort units of recovery and IX mode page set P-locks. These locks do not block access by other members, but they do block drainers such as utilities.

### ***Indoubt units of recovery (UR) with Restart Light***

When you start a DB2 with Restart Light mode and there are indoubt units of recovery, you have the following two choices:

► **Do not allow DB2 to recover the unit of recovery:**

You can release only the retained locks and then have the restarted DB2 automatically stop. The indoubt URs remain in unresolved status until connection is eventually made with the commit coordinator.

► **Allow DB2 to recover the unit of recovery:**

The restarted DB2 with light mode will be active until the last indoubt UR is resolved. If the appropriate coordinator is started on the same z/OS image, the indoubt URs will be recovered automatically. Otherwise, you have the option to recover the indoubt URs manually by issuing the RECOVER INDOUBT command.

For more information, see “Removing retained locks” in Chapter 5. “Operating with data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417.

## **6.5.1 Best practices for failures**

Make sure to perform these failure scenarios and practice recovery on a test system before going into full production.

Here are our best practice recommendations for *component failures*:

- Make sure that the SCA and Lock structures are duplexed or allocated in an external coupling facility. See “3.4, “CF configuration alternatives” on page 56”.
- Enable GBP duplexing.
- Set up SFM and ARM policies to restart failed DB2 members to release retained locks as quickly as possible.

These are our recommendations for *group restart*:

- ▶ Restart all members at the same time for fastest restart and best availability (all retained locks are released).
- ▶ While one member can perform group restart on behalf of the rest of the group, it can take significantly longer. It still leaves retained locks for the members that have not yet been started.
- ▶ Force any remaining connections or structures.

## 6.6 Commands

In this section we take a brief look at data sharing related commands. For more information, see Chapter 5. “Operating with data sharing” of *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417. We recommend that you issue certain commands to check system status on a regular schedule, preferably by using automation.

This section includes the following information:

- ▶ Basics
- ▶ CF and structure related commands
- ▶ IRLM commands
- ▶ DB2 commands

### 6.6.1 Basics

If your DB2 is installed with a command prefix scope of “S” meaning STARTED which is the default (and recommended) value, start DB2 from the z/OS image on which you want to start DB2 or you can use the z/OS console command to **route** it to another z/OS image. For the convenience of operations, you can use the route command. After DB2 is started, you can issue any DB2 commands from any z/OS image in the Parallel Sysplex group. The commands are routed to the appropriate member.

It is preferable for DB2 to be started by automation following IPLs and to be kept up by automation.

Some commands in a DB2 data sharing environment affect only one member while others cause an action that affects all members in the group. For example, **STOP DB2** is a member scope command and only affects the member which is to be stopped. **STOP DATABASE** is a group scope command and the objects within that particular database are stopped for all members. It is important that you review the *DB2 UDB for z/OS Version 8 Command Reference*, SC18-7416 for more information.

## 6.6.2 CF and structure related commands

In Table 6-1 we show a limited set of commands that display information about the CFs and their structures. Also listed are commands for starting the policy, altering the size of structures, rebuilding structures, duplexing group buffer pools, and forcing structures. Because the CFs are part of the z/OS Parallel Sysplex, the commands are z/OS commands and not DB2 commands. For more detailed information about the commands and how to use them, see *z/OS V1R7.0 MVS Setting Up a Sysplex*, SA22-7625.

Table 6-1 CF and structure related commands

Category	Command	Explanation
Starting a new CFRM policy	SETXCF START,POLICY,TYPE=CFRM, POLNAME= <i>newpolicy</i>	Activate (apply) new CFRM policy.
	SETXCF START,REALLOCATE	Resolve all pending changes from activation of a new policy, including repositioning duplexed GBPs in the preferred CF. This one command, performed by the z/OS systems staff following activation of a new policy or CF maintenance, eliminates most of the reason for the commands in “Rebuilding” and “Duplexing”.
Displaying	D XCF,STR,STRNM= <i>groupname</i> *	Display structure information.
	D XCF,STRNM= <i>strname</i> ,CONN=ALL	Display connector information in detail.
	D XCF,POLICY,TYPE=CFRM	Display the current CFRM policy.
	D CF	Display information CF and link status.
	D XCF,CF,CFNM=ALL	Display connections to the CF.
Displaying	D M=CHP	Display the status of all channel paths.
	D XCF,ARMS	Display the ARM policy.
Altering	SETXCF START,ALTER,STRNAME= <i>strname</i> , SIZE= <i>nnnn</i>	Change the size of a structure dynamically.

Category	Command	Explanation
Rebuilding	SETXCF START, RB, STRNM= <i>strname</i> , LOC=OTHER	Rebuild a structure onto another CF.
	SETXCF START, RB, STRNM= <i>strname</i> , LOC=NORMAL	Rebuild a structure based on CFRM policy.
	SETXCF START, RB, CFNM= <i>cfname</i> , LOC=OTHER	Rebuild all structures onto another CF. The <b>Reallocate</b> command is preferred.
	SETXCF START, RB, POPULATECF= <i>cfname</i>	Repopulate all structures back into a CF. The <b>Reallocate</b> command is preferred.
Duplexing	SETXCF START, RB, DUPLEX, STRNM= <i>strname</i>	Start duplexing a structure.
	SETXCF START, RB, DUPLEX, CFNAME= <i>cfname</i>	Start duplexing all eligible structures in a CF. The <b>Reallocate</b> command is preferred.
	SETXCF STOP, RB, DUPLEX, CFNAME= <i>cfname</i>	Stop duplexing all eligible structures in a CF. The <b>Reallocate</b> command is preferred.
	SETXCF STOP, RB, DUPLEX, STRNM= <i>strname</i> , KEEP=OLD NEW	Stop duplexing for a structure.
Forcing	SETXCF FORCE, CON, STRNM= <i>strname</i> , CONNM=ALL	Delete failed-persistent connections by force.
	SETXCF FORCE, STR, STRNM= <i>strname</i>	Deallocate a structure by force.

**SETXCF START, REALLOCATE** is a relatively new command that is primarily used following CF maintenance, where one CF at a time must be cleared. These activities are usually performed by the z/OS systems staff and operations. With this one command, the CFRM policy is processed sequentially for best performance and each structure is placed correctly in its preferred CF. This command eliminates the primary use of the “Rebuilding” and “Duplexing” commands that are shown on a CF basis in Table 6-1.

## 6.6.3 IRLM commands

Table 6-2 explains the IRLM modify commands. For more information about the commands, see *DB2 UDB for z/OS Version 8 Command Reference*, SC18-7416.

Table 6-2 IRLM commands

Command	Explanation
<code>F irلمproc,STATUS,ALLI</code>	Display group and member function level.
<code>F irلمproc,STATUS,ALLD</code>	Display retained locks.
<code>F irلمproc,STATUS</code>	Display waiters for locks.
<code>F irلمproc,STATUS,STOR</code>	Display the current and “high water” IRLM storage used by IRLM. <b>Recommendation:</b> Run regularly by automation to check storage status.
<code>F irلمproc,SET,CSA=nnn</code>	Change the size for MAXCSA parameter in IRLM startup procedure.
<code>F irلمproc,SET,TRACE=nnn</code>	Change the number of component trace buffers.
<code>F irلمproc,DIAG,HANG</code>	Dump all IRLMs in the group.
<code>TRACE CT,ON,COMP=irلم_name</code>	Start an IRLM trace in wrap-around mode.
<code>F irلمproc,ABEND,NODUMP</code>	Stop IRLM and de-register from ARM policy.

## 6.6.4 DB2 commands

Table 6-3 lists some of the DB2 commands that you will find useful in a data sharing environment. For more information about the commands, see *DB2 UDB for z/OS Version 8 Command Reference*, SC18-7416.

Table 6-3 DB2 commands

Command	Explanation
<code>cmd_prefix DIS GROUP (DETAIL)</code>	Display information about group in detail
<code>cmd_prefix DIS GBPOOL (gbpname)</code>	Display information about group buffer pool. <b>Recommendation:</b> Run this command regularly to monitor group buffer pool structure.

Command	Explanation
<code>cmd_prefix DIS GBPOOL(*) GDETAIL(*) TYPE(GCONN)</code>	Display statistics about <i>all</i> GBPs since the GBP was allocated. Issue it on one member only to get statistics for all members' usage of GBPs. Note: The list produced is small and contains only those GBPs that are connected at the time. <b>Recommendation:</b> Run this command regularly to monitor for "Cross Invalidations due to directory reclaims" and "Writes failed due to lack of storage"
<code>cmd_prefix DIS BP00L(bpname) LIST(*) GBPDEP(YES)</code>	Display GBP-dependent page sets for this DB2.
<code>cmd_prefix DIS DB(dbname) SPACE(*) LOCKS</code>	Display information about locks.
<code>cmd_prefix DIS DB(dbname) SPACE(*) LPL RES(LPL)</code>	Display information about LPL entries.
<code>cmd_prefix DIS DB(dbname) SPACE(*) RES</code>	Display information about objects in restricted status.
<code>cmd_prefix STA DB(dbname) SPACE(spacename)</code>	Recover objects in LPL or GRECP status.

## 6.7 Multi-system DB2 diagnostic dumps

In a DB2 data sharing environment, it is possible for one or more members to experience problems or even completely fail due to an abnormal situation on a different member within the group. For problem determination, IBM defect support may require dumps of particular address spaces from *all of the members* of the data sharing group.

Here are our general recommendations for dealing with a "hang" of all the members in the data sharing group:

- ▶ Dump all DB2 address spaces on all members.
- ▶ Dump all IRLM address spaces and include XES information for all members:
  - Provide SYS1.LOGREC and SYSLOG for all member from about one hour before the hang to the time of the dumps
  - If you can identify the home address space of any job involved in the hang, its dump would be an added benefit (however, this may not be possible).

- Set up an SLIP for taking any dump, using the coding in Example 6-2.

*Example 6-2 Setting parameters for dumps*

---

```
SLIP SET,IF,N=(IEAVEDSO,00,FF),A=SVCD,DISABLE,  
      JOBLIST=(XCF*,????IRLM,???MSTR,???DBM1),  
      SDATA=(XESDATA,COUPLE,PSA,LSQA,LPA,GRSQ,SWA,RGN,CSA,  
            SQA,SUM,TRT,ALLNUC),  
      REMOTE=(JOBLIST,SDATA),ID=xxxx,END
```

where ???? is the high level SSID name of the members.

WHERE XCF\* = your XCF name

WHERE xxxx is your slip id that you can use to  
enable the slip when the error occurs.

---

When you set the SLIP, it will be set as DISABLED. When you want a dump, just issue the following command and the dump will be taken at that time:

```
SLIP MOD,ENABLE,ID=xxxx
```

## 6.8 Disaster recovery

It is critical for enterprises to protect their DB2 for z/OS data in case of disaster and to be able to restart with a consistent copy of the DB2 data as quickly as possible with minimal loss of data. The implementation of DB2 data sharing introduces some additional activities that are required both in the preparation for and execution of the disaster recovery procedure.

The redbook *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370, provides information on data sharing disaster recovery requirements as well as those for non-data sharing.

## 6.9 Rolling maintenance

Applying maintenance to a DB2 subsystem generally requires a scheduled outage. With a data sharing group, such outages can be avoided by using a technique called rolling maintenance.

Workload is shifted from one member to another member. This implies that the applications must be able to run on more than one member. One DB2 member of the data sharing group is stopped while the other members keep processing the workload. The system programmer makes appropriate modifications so that when that DB2 is started again, it loads modules that have the new maintenance applied. The process is repeated for each of the other members, usually one at a time, until all the members are running with the new maintenance.

Rolling maintenance across all the members maintains data and application availability with no apparent outage to end users.

### 6.9.1 Service recommendations

Use consolidated service test (CST) or recommended service upgrade tape (RSU) maintenance packages for all main IBM products, as this service has been tested together and should be more resilient than independent problem temporary fixes (PTF). After 6 months 60% of errors identified by customers had PTFs. Putting on a fix for a critical problem if you are far behind may require you to apply many PTFs that are in the prerequisites chain into production.

Staying between 3 and 6 months behind currency reduces the chances of encountering a PTF in error (PE). Most PEs are discovered within the first 3 months of PTF close date. This level of currency also reduces the chance of encountering a known problem for which a fix already exists but has not been applied in the environment

ERRSYSMOD report warns of fixes that are going PE, new high pervasive APAR (HIPER) / Security / Integrity / Pervasive PTFs available. If you know you are exposed to a problem with a critical fix, implement it quickly and do not wait for the next maintenance cycle.

For a better description of the Consolidated Service Test (CST) process, go to:  
<http://www.ibm.com/servers/eserver/zseries/zos/servicetst/>

### 6.9.2 Best practices for maintenance

Here are our best practice recommendations for maintenance:

- ▶ Apply CST/RSU maintenance minimum of 2-3 times a year. Stay no less than 3 months and no more than 6 months behind in production.
- ▶ Download HIPERs weekly and apply them monthly.
- ▶ Test maintenance to assure stability in your environment.





## Advanced topics

This chapter provides information for additional data sharing issues that might arise. For more details on them, you should consult the referenced publications. Each topic stands on its own and may not bear any relationship to any other topic in this section.

We discuss the following topics:

- ▶ CLOSE YES and CLOSE NO table spaces
- ▶ The problem of runaway threads
- ▶ Lock avoidance: CLSN versus GCLSN
- ▶ Usermod for routing to multiple DB2s on a z/OS image
- ▶ Performance
- ▶ Determining the number of threads

## 7.1 CLOSE YES and CLOSE NO table spaces

The CLOSE specification you made for CREATE/ALTER TABLESPACE has different implications in data sharing.

Any table space or index which has remained in a read-only state with no activity for a period of time longer than the pseudo-close interval (PCLOSEN/PCLOSET DSNZPARM) is physically closed to reduce data sharing overhead (the object can become non-GBP-dependent if it is closed on all but one member or if it is open on multiple members but all are read-only). Its disadvantage is that the next access to the object must go through physical open again. There can be significant overhead, particularly if a large number of objects are accessed at the same time after a period of inactivity (such as beginning of day).

In data sharing, DB2 honors the CLOSE YES or NO attribute of the table space or index. The physical close now takes place only for CLOSE YES objects, while CLOSE NO objects remain open indefinitely. You control the behavior on an object-level basis directly instead of through PCLOSEN/PCLOSET DSNZPARM values, because lengthening the pseudo-close interval has adverse effects on recovery.

In all cases pseudo-close still occurs. The ending SYSLGRNX entry is closed, thereby delimiting for the RECOVER utility the location of updates. Additionally the HPGRBRBA in the header page is updated for each page set (except for the seven always open Catalog/Directory table spaces).

There are certain performance considerations to be evaluated:

- ▶ CLOSE (YES): After a pseudo-close interval with no other interest, the data set is physically closed and can experience the performance degradation described previously, as a physical open must be performed on next access. For objects that are infrequently GBP-dependent, CLOSE(YES) likely provides better performance and faster restart after a DB2 crash.
- ▶ CLOSE (NO): Data set is not closed. This means that once an object becomes GBP dependent, it will remain so until the object is pseudo-closed as the only read/write member. Pages will continue to be registered and changed pages will be written to the GBP and subsequently castout *even if there has been no other interest for hours or days*. This can have an effect on the amount of data sharing overhead experienced in those cases.

### 7.1.1 Best practice

Specify CLOSE(NO) for objects that you expect to be GBP-dependent most of the time and which are sensitive to response time during first access. Most table spaces should be CLOSE(YES).

## 7.2 Performance

The question everyone asks is: “How much overhead will I experience in data sharing over my current non-shared workload?”

That answer cannot be provided here, as it is workload dependent. The most accurate way to determine it is to provide an isolated environment with repeatable workload where you can benchmark, but this is impractical. New data sharing users are left with commonly accepted rules of thumb (ROT) that are stated something like, “For a 2-way group, expect overhead of x% and for each additional member, y%.” The y% value is typically <1%. The numbers given originally in the mid 1990s were based on 9672 technology and traditional OLTP-like workloads.

Today’s DB2 user has a much more diverse workload, one that is less repeatable, more distributed, and uses much faster System z processors with internal coupling facilities. The old numbers are based on DB2 V4. There have been numerous improvements in DB2 since that time to reduce data sharing overhead. The most significant ones have occurred in DB2 V8, which reduces global locking contention, while it increases batching capability for CF access.

What we can say is that you will achieve the best performance when you adopt the recommendations we describe in this document:

- ▶ Sizing the lock structure to eliminate false contention. See 7.3, “Eliminating false contention” on page 124.
- ▶ Sizing the GBPs to eliminate “cross invalidations due to directory reclaims”. See 3.1.1, “Group buffer pools” on page 40.
- ▶ Implementing auto alter to insure an optimal directory to data ratio for the GBP. See 3.2, “Auto Alter” on page 47.
- ▶ Providing two dedicated links between the z/OS image and each CF to reduce or eliminate path busy conditions
- ▶ Providing at least two CF engines per coupling facility for most predictable response. See 7.4, “How many CF engines are necessary?” on page 126.
- ▶ Committing frequently in batch jobs to allow effective lock avoidance. See 7.5, “Lock avoidance: CLSN versus GCLSN” on page 128.

- ▶ CF technology that is no more than one generation behind the processors on which z/OS reside. See “Technology considerations for external CFs” on page 60.
- ▶ Minimize the GBP-dependency of DB2 objects through scheduling or direction of workload. This negatively affects the availability afforded when an application can be routed to any and all members of a group.

## 7.3 Eliminating false contention

The discussion in this section assumes, at a minimum, all of these conditions:

- ▶ DB2 V8
- ▶ New Function Mode (NFM)
- ▶ Group stop/start to implement lock protocol 2

In DB2 V8 NFM, once you have restated the data sharing group after a group quiesce, you have automatically enabled protocol 2. This protocol avoids the cost of global contention processing whenever possible. It also improves availability due to a reduction in retained locks following a DB2 subsystem or z/OS system failure making the recommendation for `RELEASE(DEALLOCATE)` (and thread reuse) to reduce XES messaging for L-locks no longer required. For details, see Chapter 8, “Data sharing enhancements” of the redbook *DB2 for z/OS Version 8 Performance Topics*, SG24-6465.

We have recommended you size your lock structure initially at 64000K. How do you know if this sizing is correct after you have been in data sharing for awhile? While we recommend you refer to the *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417 for more detailed performance and tuning assistance, we discuss in this section how to minimize the false contention (FC) that can occur when two different objects (*a* and *b*) hash to the same lock table entry (LTE).

When there is false contention on an LTE, the XES component on one member must resolve it by messaging to a XES component on the other member. The transaction is suspended while the resolution is accomplished. It is always successfully resolved by XES without taking the further step of consulting the IRLMs involved because XES recognizes that *a* and *b* are different.

**Important:** No matter how many members exist, *no more than one other member* need be messaged to resolve global lock contention. This means that for a 6-way data sharing group, it is not necessary for all 6 members to vote if contention exists. DB2 does not perform pass-the-buck global locking.

The idea behind tuning to minimize false contention is that if the lock table contained more entries, *a* and *b* would hash to different LTEs and lock requests would be granted without suspension.

We use the RMF coupling facility activity report to determine quickly if the structure is sized correctly. This report is normally produced by the z/OS parallel sysplex team. Example 7-1 is a part of the report with statistics for the lock structure

Example 7-1 Snippet of RMF CF activity report for the DB2 lock structure

1	COUPLING FACILITY ACTIVITY													PAGE 30	
z/OS V1R6		SYSPLEX PRODPLEX				START 08/10/2006-08.00.00				INTERVAL 001.00.00					
		RPT VERSION V1R5 RMF				END 08/10/2006-09.00.00				CYCLE 01.000 SECONDS					
-----															
COUPLING FACILITY NAME = CF1															
-----															
COUPLING FACILITY STRUCTURE ACTIVITY															
-----															
STRUCTURE NAME = DSNDBPO_LOCK1 TYPE = LOCK STATUS = ACTIVE															
# REQ		-----		REQUESTS		-----		DELAYED REQUESTS		-----					
SYSTEM	TOTAL	#	% OF	-SERV	TIME(MIC)	-	REASON	#	% OF	----	AVG	TIME(MIC)	----	EXTERNAL REQUEST	
NAME	AVG/SEC	REQ	ALL	AVG	STD_DEV			REQ	REQ	/DEL	STD_DEV	/ALL		CONTENTIONS	
...															
... Report lines removed															
SYS1	137M	SYNC	137M	81.0	16.5	21.0	NO SCH	59K	0.0	90.7	379.3	0.0	REQ TOTAL	121M	
	38033	ASYN	409K	0.2	157.7	219.8	PR WT	0	0.0	0.0	0.0	0.0	REQ DEFERRED	473K	
		CHNGD	0	0.0	INCLUDED	IN ASYN	PR CMP	0	0.0	0.0	0.0	0.0	-CONT	473K	
													-FALSE CONT	309K	
...															
SYS2	31663K	SYNC	32M	18.7	18.0	33.3	NO SCH	18K	0.1	153.9	425.9	0.1	REQ TOTAL	26M	
	8795	ASYN	136K	0.1	189.2	227.2	PR WT	0	0.0	0.0	0.0	0.0	REQ DEFERRED	370K	
		CHNGD	0	0.0	INCLUDED	IN ASYN	PR CMP	0	0.0	0.0	0.0	0.0	-CONT	369K	
													-FALSE CONT	217K	
... Report lines removed															
-----															
TOTAL	169M	SYNC	168M	100	16.8	23.8	NO SCH	77K	0.0	105.7	391.8	0.0	REQ TOTAL	148M	
	46828	ASYN	545K	0.3	165.6	222.1	PR WT	0	0.0	0.0	0.0	0.0	REQ DEFERRED	843K	
		CHNGD	0	0.0			PR CMP	0	0.0	0.0	0.0	0.0	-CONT	842K	
													-FALSE CONT	526K	

Example 7-1 shows the part of an RMF CF activity report that identifies the lock structure of a 2-way data sharing group. The fields to look at are those in bold in the lower right corner of the report titled:

REQ TOTAL 148M  
REQ DEFERRED 843K  
-CONT 842K  
-FALSE CONT 526K

The total number of lock requests during this interval was 148M. Of those 526K were for false contention. Our false contention is 526K/148M for .3%. The current size of this lock structure is correct. We expect some amount of false contention.

If false contention exceeds 1%, we would recommend increasing the LTEs to the next power of 2.

REQ DEFFERED shows how many lock requests were deferred, usually because of contention. The -CONT field of 842K identifies all contention, whether it is for:

- ▶ FC contention, where two different objects hash to the same LTE
- ▶ XES contention, where the same table is open with an S mode Logical lock (L-lock)
- ▶ IRLM contention, where two different members want an exclusive lock on the same page

From RMF you can identify only global contention (REQ DEFERRED) and FALSE CONT. For IRLM and XES contention you need to look at the SUSPENDS - XES GLOBAL CONT and SUSPENDS - IRLM GLOBAL CONT fields in the OMEGAMON XE for DB2 PM/PE Statistics Long report.

For best performance the global lock contention should be less than 5%. If it exceeds 5% and is attributed to IRLM contention, the only way to reduce it is through changes to the application SQL statements.

Most of the time REQ DEFERRED is the same value as -CONT, but it can be different when there is a deferral for certain other internal activities, such as XES latch contentions.

Another anomaly is the difference between the total requests on the left hand side, which are less than those on the right hand side. The total requests value on the left side is actual CF requests. The total requests value on the right side is *logical* requests. For the most part these are 1:1, but not necessarily. The classic example is *batch unlock* processing, where 1 CF request on the left could handle more than 1 unlock request (as counted on the right).

## 7.4 How many CF engines are necessary?

DB2 V8 introduced new CF instructions that handle multiple CF requests at one time. They are Read For Castout Multiple (RFCOM) and Write and Register Multiple (WARM). The intent of these instructions is to take a number of individual instructions and lump them into one instruction that provides a list of actions to be performed. WARM allows multiple pages to be written to the CF and registered with a single write request. The intent is to take a CF request that would take 30 µsec. synchronous service time (about 10 µsec. on the CF CPU) and combine may be 10 of them into one instruction that executes in 100 µsec. on the CF CPU. CF utilization remains the same but host and link activity is reduced, especially in DB2 batch insert/update applications.

**Note:** The numbers used in this section are for illustrative purpose only.

The new instructions contribute to significant increases in asynchronous requests, as synchronous requests are converted to asynchronous most of the time due to the increased time to process Multiple requests.

**Single CF engine effect:** Assume there are two CF requests that arrive about the same time. The first request will consume 100  $\mu$ sec. on the CF itself (and is likely asynchronous). The second request will consume 10  $\mu$ sec. on the CF and is synchronous. The synchronous request must wait on the only CF processor (not a subchannel) until the first request completes. If there had been a second CF engine, it could have processed the synchronous request immediately. The result of this activity on a CF with a single engine is to have *variable response times* that are reflected in the RMF CF Activity Report in the STD\_DEV column under *SERV TIME (MICS)* and DELAY /ALL.

The longer running commands have the side-effect of monopolizing the single CF CPU while they are running, so that any other single-entry CF requests (Register Page, for example) that come through will have to wait behind them. This will tend to elongate the *SERV TIME (MICS)* sync service time of the single-entry requests and increase their standard deviation as well. If it happens to a very significant extent, it could increase the average service time for even single-entry commands to the point where they also start to get converted to async. When your peak CF CPU utilization on a single engine CF approaches 25%, we recommend you add a second engine.

For this reason alone, we recommend that you follow these suggestions:

- ▶ Do not share CF engines in a production parallel sysplex.
- ▶ Consider providing two dedicated engines minimum per CF for a production parallel sysplex for consistent service times.

For less-than-optimal configurations, our guidelines are as follows:

- ▶ Do not share CF engines in a production parallel sysplex (test and development environments have less stringent requirements)
- ▶ If you currently have a single CF engine (per CF), add a second when peak CF CPU utilization exceeds 25% (single engine effect).

Look at the RMF CF activity report, under the coupling facility usage summary and find the number beside *Average CF utilization (% busy)*

- ▶ If you have a multi-engine CF, add another engine when peak CF CPU utilization exceeds 50%.

These guidelines also apply if you implement system-managed duplexing:

- ▶ If you currently have a single CF engine per CF, add a second engine when peak CF CPU utilization exceeds 25%.
- ▶ If you have a multi engine CF, add another engine when peak CF CPU utilization exceeds 50%.

## 7.5 Lock avoidance: CLSN versus GCLSN

Lock avoidance in a data sharing environment uses the same techniques as those for non-shared DB2s. The only new concept added for this environment is the global commit log sequence number (GCLSN). For non-GBP-dependent page sets or partitions, each member uses a local commit log sequence number (CLSN), which is the lowest uncommitted log sequence number for all active transactions for a given page set.

For GBP-dependent page sets, a single GCLSN value is maintained and used for lock avoidance for the entire data sharing group. The single GCLSN value is derived from each member's CLSN values. The member's CLSN value is maintained at the DB2 subsystem level, not at the page set or partition level. It is the beginning of unit of recovery (UR) LRSN value of the oldest UR in this DB2 member.

The GCLSN is the earliest CLSN value across all members across all page sets or partitions, regardless of whether the page sets or partitions are GBP-dependent or not. Each member independently updates the GCLSN periodically, based on timer interval. Periodically, each member pushes the latest CLSN value into the SCA and pulls from the SCA the lowest value across all members.

One long-running update process without intermediate commit points can effectively stop the GCLSN value from moving forward regardless of whether the process is updating GBP-dependent or non-GBP-dependent objects. Because this value is low, the CLSN technique qualifies fewer pages for lock avoidance. also see 7.6, "The problem of runaway threads" on page 128.

## 7.6 The problem of runaway threads

The problem is pervasive to all DB2 applications with or without data sharing. A "minor" application may be developed and tested successfully and moved into production. Once there the "minor" application is enhanced to process more



data, access other tables, and send results to other applications to process. It now becomes a critical application, but its basic application characteristics remain as they were when it was developed. The amount of data becomes much larger than the original scope and the application, which doesn't commit its work, runs longer, perhaps an hour, before ending. If it abends after 50 minutes, it is likely to take 100 minutes to roll back, consuming resources as it "undoes" its 50 minutes of updates.

These are the issues concerning runaway jobs:

- ▶ An online system has high concurrency requirements and long running updaters prevent concurrency either by taking many locks until the job finishes or by escalating page/row locks into exclusive table space locks and preventing all access until the job commits or completes.
- ▶ Lock avoidance techniques fail, resulting in increased locking rates, increased CPU and sometimes increased response time. Lock avoidance is enhanced for packages bound with CURSOR STABILITY and CURRENTDATA(NO). When a long running updater has been running an hour or more, many more locks are taken by all other users in the entire data sharing group. When in the past a requestor could read an entire page, examining every row without a lock, now chances are good that a lock must be taken. See 7.5, "Lock avoidance: CLSN versus GCLSN" on page 128.
- ▶ Utilities cannot break in. They may not be able to break in via the DRAIN lock, as claims by non-utilities are not released until commit. While updaters hurt concurrency far more than readers, readers that do not commit can cause Online Reorg to fail in the Switch phase.
- ▶ Resources are wasted if the job abends. Back out consumes many resources and all work is thrown away.
- ▶ Even with the improvements in DB2 restart such that a long running job backing out does not prevent new work, the data involved in the back out are unavailable to new work until back out is complete.
- ▶ A long running updater continues to acquire thread storage in the DBM1 address space,
- ▶ A long running update acquires many locks and can stress the RLE part of the lock structure.

Customers have wanted DB2 to tell them when this situation is beginning, so that they can prevent the problem by canceling the job. DB2 provides warning messages when user-established DSNZPARM thresholds are exceeded (DSNR035I for URCHKTH and DSNJ031I for URLGWTH).

What action can be taken other than to cancel the job? This is a drastic measure if the application is critical. Once messages are issued, the application should be

coerced into incorporating a reasonable commit scope. This is hard to do and is usually done as part of a long term project that must have full management support from both the operations and application perspectives.

One approach is to set the URCHKTH and URLGWTH higher in the beginning, gather the messages issued and request the offenders to write commit logic in their programs. What would be high values in a fast processor? If CHKFREQ is 3 minutes, then set URCHKTH to 15, which would mean that a UR had gone 45 minutes without a commit, an extremely long time. The goal would be to reduce it over time to 3 times the CHKFREQ. Similarly, set URLGWTH to 100 (100,000 updates) and reduce it over time to a goal that the business can support.

The DSNZPARM LRDRTHLD is used to identify readers that do not commit (in number of minutes). IFCID313 is produced when readers exceed this threshold, but no DB2 message is issued.

## 7.7 Usermod for routing to multiple DB2s on a z/OS image

The situation described in this section is the reason we do not want you to have a member with the same name as the group attach name.

### 7.7.1 Multiple DB2 members in one z/OS image

As long as the connection is made to a DB2 member by its subsystem name (SSID), each DB2 receives the work intended for it.

There is great flexibility enabled by the specification of the group attach name for various connections (TSO, utilities, and others). If connection is made through specification of the group attach, it can run on any z/OS image that has a member of the group. If there are multiple members of the same z/OS, then it will be routed to one of the members only. The IEFSSNxx member of SYS1.PARMLIB is searched for the first active DB2 with the specific group attache, but it is not possible to predict which one will receive the traffic as the IEFSSNxx member used is hashed.

With DB2 V8, if you want to *distribute workload among two DB2 members on a single z/OS image*, you can request a usermod from the IBM Support Center through DB2 Level 2. Because there are restrictions on its use, you must contact the IBM Support Center directly. General restrictions are:

- ▶ You cannot have a member with the same name as the group attach.
- ▶ You cannot use ODBC.

- You cannot use JDBC Type 2 driver connectivity and specify the group attach name.

The JDBC T2 driver does not use the NOGROUP option to connect. So if MemberName = GroupName then this name would be treated as a Group Name in some cases. If you are not using JDBC Type 2 driver connectivity or if JDBC connections always use member name and it is not the same as the group name, then the usermod can be used.

For DB2 Version 9.1 for z/OS, no usermod is necessary.

**Note:** In order to have multiple DB2 members on the same z/OS image at all, you must specify TCP/IP SHAREPORT for the SQL listening port (not for the RESPORT).

## 7.8 Determining the number of threads

You should read this section when you are trying to determine if you need to add another DB2 member.

In DB2 V8 the maximum number of concurrent threads is 2000 and must be apportioned between CTHREAD and MAXDBAT in DSNZPARM. The actual number of concurrent threads that can be supported in the *ssss*DBM1 address space is commonly many fewer than that.

DB2 needs a certain minimum virtual storage, called the storage cushion in order to stay up. If the amount of storage available remaining in the DBM1 address space goes below this threshold, DB2 triggers a process to free storage. This process has a considerable CPU overhead and can be very disruptive for service. If the storage continues to drop DB2 will cancel existing work (E20003/E20016 abends). If it cannot recover enough storage from these actions, DB2 may fail in order to protect the integrity of its data. The storage cushion size is determined by the MVS Extended Region Size (EPVT) and three DSNZPARM values: the maximum number of open data sets (DSMAX); the maximum number of allocated threads (CTHREAD); and the maximum number of distributed threads (MAXDBAT).

One of the most frequent problems today in large DB2 subsystems is lack of sufficient virtual storage. Usual causes are widespread usage of dynamic SQL with the dynamic statement cache, large numbers of threads, both CTHREAD and MAXDBAT, and a large number of open data sets (DSMAX). Proper management of DB2 virtual storage will prevent E20003/E20016 abends and subsequent abnormal termination of DB2.

## 7.8.1 REXX tools package

The most accurate way to determine the maximum number of threads is to use a package consisting of REXX execs distributed on an as-is basis on the DB2 for z/OS Examples Trading Post at

<http://www.ibm.com/software/data/db2/zos/exHome.html>

Look for it under the *REXX Execs* topic. It is called “REXX tools package to retrieve storage information.” It is commonly called *MEMU* or *MEMU2*.

It consists of the following files that are zipped:

- ▶ MEMU2 REXX – outputs IFCID225 information invoked as batch job
- ▶ MEMUSAGE REXX – outputs IFCID225 (now) if invoked from TSO Option 6 (once)
- ▶ Memu2.jcl.txt – Sample JCL to invoke MEMU2 REXX as a batch job
- ▶ Memory Usage-V3-IFCID225.doc – documentation to install, modify, and use

MEMU2 REXX is

- ▶ Invoked via JCL on single member basis
- ▶ Outputs IFCID225 information to comma delimited data set
- ▶ at 5 minute interval (default) independent of STATIME
- ▶ for 12 intervals default (one hour)
- ▶ JCL specifies SSID and overrides to above

The primary exec, MEMU2, runs as a simple JCL job that writes the DBM1 virtual storage trace, IFCID225, to a comma delimited file. It can run for as long an interval as you wish. To get an accurate picture of virtual storage characteristics, you should run MEMU2 for long enough to identify peak virtual storage periods, such as heavy first of month or end of month activity, weekend utility jobs or other periods where you suspect virtual storage demand may be heavy.

The goal of this process is to determine the correct value for maximum number of threads that can be supported with the workload measured. Once you have identified the value, you should divide it between CTHREAD and MAXDBAT.

There is a teleconference presentation available, of which a section about virtual storage tells you how to use the MEMU2 output to determine the maximum number of threads. It is expected to be available until July 2008. You can register for the replay at

<http://www.ibm.com/software/zseries/telecon/25jul>

Once registered, you will be sent an E-mail with the Web site for the MP3 replay and the presentation.

See the redbook *DB2 for z/OS Version 8 Performance Topics*, SG24-6465, Section 4.3, for a more thorough discussion of DBM1 Virtual Storage calculations.

Periodically review the informational APAR II10817 to ensure you have applied any service that affects DB2 storage.

## 7.8.2 Statistics Spreadsheet support

An alternative to MEMU is available for users of IBM Tivoli OMEGAMON XE for DB2 Performance Expert on z/OS. PK31073 (PTF UK17906) introduces the SPREADSHEETDD subcommand for the STATISTICS FILE. This option allows you to extract virtual storage data in data sets with comma-separated values. This format can be easily imported into workstation-based spreadsheet programs. You can then use spreadsheet functions, such as sorting and filtering, to evaluate the data.

The output file is similar to that produced by MEMU, but its data is produced only at the statistics interval STATIME.

A description of the new function is described in a document entitled “*SPREADSHEETDD subcommand option for STATISTICS FILE*”, which is available at:

<http://www.ibm.com/support/docview.wss?uid=swg27008487>





## Best practices

In this chapter we summarize the best practices that we have mentioned in the preceding chapters of this book.

As stated in “Our assumptions” on page ix, we are assuming that you are an experienced DB2 user who is working in DB2 V8 New Function Mode, that you already have a Parallel Sysplex, and that you are going to implement two way data sharing for the first time.

Most of our recommendations in this list are specifically intended for the new data sharing user.

Experienced data sharing users may want to evaluate their environments as well using this list.

Most of the DB2 considerations are applicable to DB2 Version 7 environments as well, with the exceptions noted in “Our assumptions” on page ix.

## 8.1 Table of best practice recommendations

Table 8-1 shows the *best practices for data sharing*. The left hand column contains a recommendation, and the right hand column contains a reference to where that topic is discussed in this book.

Table 8-1 *Best practices for data sharing*

Best practices for getting started	Where discussed in this book
Keep the DB2 subsystem name of your non-data sharing member as your group attach name.	4.1.1, "Group name" on page 66
Do not name a DB2 member with the same SSID the same as your group attach name.	4.1.1, "Group name" on page 66
Make the SSID part of your log and BSDS data set names. Include the date/time in the archive data set name.	4.1.4, "Log data set names" on page 68 and 4.1.5, "Bootstrap data set (BSDS)" on page 68
For availability during for planned outages, ensure the application can run on more than one z/OS image (at least two DB2 members)	2.5.1, "Portability" on page 36
Implement location aliases to avoid making changes to your applications coming from other DB2 subsystems.	4.1.6, "Distributed Data Facility (DDF) related parameters" on page 69
Make sure that active log copy 1 is on a different disk from that of active log copy 2.	4.2.1, "Active log data sets" on page 73
Archive to disk and offload to tape after 24-48 hours or in line with your image copy cycle. Do not place archive data sets from different members on the same physical tape,	4.2.2, "Archive log" on page 74
Use CFSizer to size the GBPs.	3.1.1, "Group buffer pools" on page 40
Start with 64,000 for lock structure size.	3.1.2, "Sizing the lock structure" on page 46
Start with 64,000 for SCA structure size.	3.1.3, "Sizing the SCA structure" on page 47
Implement Auto Alter for the GBPs as long as enough storage exists in the CFs for the GBPs to be correctly sized.	3.2.1, "Implementing Auto Alter" on page 50
Make sure one CF can take over the workload in both CFs in terms of CF storage.	3.2.1, "Implementing Auto Alter" on page 50
Ensure all DB2 structures rebuild into another CF upon any failure.	3.4, "CF configuration alternatives" on page 56



<b>Best practices for getting started</b>	<b>Where discussed in this book</b>
Duplex the GBPs specifying DUPLEX (ENABLED).	3.3.1, "Group buffer pools (GBP) or user-managed pools" on page 54 and 6.2.3, "CF failure" on page 111
Guard against double failure in a Parallel Sysplex through use of an external CF for the Lock and SCA structures and an ICF.	3.4.1, "ICF-only: Double failure for the lock and SCA structures" on page 57
Provide 2 links from each image to each CF.	3.3.1, "Group buffer pools (GBP) or user-managed pools" on page 54
Match an external CF's technology to the processor.	"Technology considerations for external CFs" on page 60
Provide 2 dedicated engines minimum per CF for a production Parallel Sysplex. Add another, when peak CF CPU utilization exceeds 50%.  If each CF is a single engine, add a second engine when the peak CF CPU utilization approaches 25%.	7.4, "How many CF engines are necessary?" on page 126
Set GBP thresholds <ul style="list-style-type: none"> <li>– GBP Checkpoint 4 minutes</li> <li>– GBPOOLT 10-30%</li> <li>– CLASST 2-5%</li> </ul>	"GBP thresholds" on page 52
Use Dynamic virtual IP addressing (DVIPA) for availability and failover.	5.3, "Dynamic virtual IP addressing (DVIPA)" on page 94
Share SYS1.PROCLIB among the members of the data sharing group and use one copy of the JCL for each stored procedure.	5.6, "Stored procedures" on page 97
Ensure all your DB2 stored procedures run in WLM-established stored procedure address spaces.	5.6, "Stored procedures" on page 97
For DB2 for z/OS requesters to a data sharing group, use TCP/IP connectivity.	5.5, "Distributed data facility (DDF)" on page 95
If you still use an SNA network, use member routing for requests from a DB2 for z/OS requester to a data sharing group.	5.5, "Distributed data facility (DDF)" on page 95
Apply CST/RSU maintenance 2-3 times a year. Stay no less than 3 months and no more than 6 months behind in production	6.9.1, "Service recommendations" on page 120
For any manual GRECP/LPL recovery start databases in priority sequence using a list. Enable fast log apply.	6.1.4, "Recommendations for GRECP/LPL recovery" on page 107
Do not issue start commands for each individual object for manual GRECP/LPL recovery.	6.1.4, "Recommendations for GRECP/LPL recovery" on page 107

Best practices for getting started	Where discussed in this book
Use existing automation or ARM for restarts in place.	6.4, “Automatic restart manager (ARM)” on page 112
Have the z/OS systems team Implement an ARM policy for cross system restart to release retained locks as quickly as possible	6.2, “Component failure” on page 109, “ICF-only: Double failure for the lock and SCA structures” on page 57 and “Automatic restart manager (ARM)” on page 112
Restart <i>all</i> members at the same time for group restart	6.2, “Component failure” on page 109
Implement a naming convention that is easy for operators and automation routines. <ul style="list-style-type: none"> <li>► Good: DBP1, DBP2, DBP3</li> <li>► Bad: ABCD, DBP1, DBP2</li> </ul>	4.1, “Naming conventions” on page 66
Rename your existing non-shared DB2 subsystem before you enable data sharing.	4.4, “Renaming an existing non-data sharing member” on page 78
Activate your data sharing related CFRM policy before enabling data sharing,	4.5, “Enabling the data sharing group” on page 85
Have the z/OS systems team implement SFM policy with ISOLATETIME parameter.	6.3, “Sysplex failure management (SFM) policy” on page 112
Keep false contention <1% of total Global Locks.	7.3, “Eliminating false contention” on page 124
Specify CLOSE(NO) for objects that you expect to be GBP-dependent most of the time and which are sensitive to response time during first access. Most table spaces should be CLOSE(YES).	7.1, “CLOSE YES and CLOSE NO table spaces” on page 122
Ensure batch or other updaters commit work frequently according to a business-defined scope.	7.6, “The problem of runaway threads” on page 128 and 7.5, “Lock avoidance: CLSN versus GCLSN” on page 128

# 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 142. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *DB2 for z/OS Version 8 Performance Topics*, SG24-6465
- ▶ *Disaster Recovery with DB2 UDB for z/OS*, SG24-6370
- ▶ *Parallel Sysplex Application Considerations*, SG24-6523
- ▶ *Achieving the Highest Levels of Parallel Sysplex Availability in a DB2 Environment*, REDP-3960
- ▶ *Use of Shared Engines for Coupling Facilities - Implementation and Impact*, TIPS0237
- ▶ *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*, SG24-7083
- ▶ *DB2 UDB for z/OS Version 8: Everything You ever Wanted to Know,... and More*, SG24-6079
- ▶ *System Programmer's Guide to: Workload Manager*, SG24-6472
- ▶ *TCP/IP in a Sysplex*, SG24-5235

## Other publications

These publications are also relevant as further information sources:

- ▶ *DB2 for z/OS Version 8 Data Sharing: Planning and Administration*, SC18-7417-03
- ▶ *DB2 UDB for z/OS Version 8 Utility Guide and Reference*, SC18-7427-02
- ▶ *DB2 UDB for z/OS Version 8 Installation Guide*, GC18-7418-03
- ▶ *DB2 UDB for z/OS Version 8 Command Reference*, SC18-7416-03
- ▶ *z/OS DFSMSdfp Storage Administration Reference*, SC26-7402-05

- ▶ *DB2 for z/OS Application Programming Guide and Reference for Java™*, SC18-7414
- ▶ *z/OS V1R7.0 MVS Setting Up a Sysplex*, SA22-7625-12
- ▶ *z/OS V1R7.0 MVS System Commands*, SA22-7627-13
- ▶ *z/OS MVS Planning: Workload Management*, SA22-7602
- ▶ *Leveraging z/OS TCP/IP Dynamic VIPAs and Sysplex Distributor for higher availability*, GM13-0165
- ▶ *Coupling Facility Performance: A Real World Perspective*, REDP-4014
- ▶ White Paper entitled “*Coupling Facility Configuration Options*,” available on the Web, at:  
<http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gf225042.pdf>
- ▶ *System-Managed Coupling Facility Structure Duplexing*, GM13-0103, at:  
<http://www.ibm.com/servers/eserver/zseries/library/techpapers/gm130103.html>
- ▶ White paper called *System-Managed CF Structure Duplexing Implementation Summary*, at:  
<http://www.ibm.com/servers/eserver/zseries/library/techpapers/pdf/gm130540.pdf>

## Online resources

These Web sites are also relevant as further information sources:

- ▶ Parallel sysplex technology and benefits:  
<http://www.ibm.com/systems/z/resiliency/parsys.html>
- ▶ Getting started with Parallel Sysplex documents:  
<http://www.ibm.com/servers/eserver/zseries/zos/bkserv/zswpdf/zospsysgstart.html>
- ▶ System z Server Time Protocol (STP):  
<http://www.ibm.com/servers/eserver/zseries/pso/stp.html>
- ▶ Web site for WLM home page:  
<http://www.ibm.com/servers/eserver/zseries/zos/wlm/>
- ▶ CFSIZER can be found at:  
<http://www.ibm.com/servers/eserver/zseries/cfsizer/>

- ▶ Consolidated Service Test (CST) process:

<http://www.ibm.com/servers/eserver/zseries/zos/servicetst/>

- ▶ DB2 for z/OS Examples Trading Post can be found at:

<http://www.ibm.com/software/data/db2/zos/exHome.html>

Look under the *REXX Execs* topic for “REXX tools package to retrieve storage information.” The virtual storage EXEC is commonly called *MEMU* or *MEMU2*.

- ▶ Teleconference presentations:

There are two teleconference presentations available for data sharing, presented by the authors of this book. The recordings and presentations are expected to be available until July 2008. You can register for each replay and you will be sent an E-mail with the Web site for the MP3 replay and the presentation.

- “Why DB2 for z/OS data sharing should be in your life...” is directed at the reader of this book and covers the following topics:

- Why are customers doing data sharing?
- How much overhead can you expect from data sharing?
- How long does it take to implement data sharing?
- Do my applications have to change when I implement data sharing?
- Does data sharing allow for parallelism?

<http://www.ibm.com/software/zseries/telecon/11jul>

- “Data Sharing Health Checks: what we have learned” is directed at current data sharing users and covers the following topics:

- How many directory entries do I really need for my Group Buffer Pools...how do I handle GBP growth as my data sharing group adds new members...and what changes should I make to GBPs as each member's local buffer pools change?
- How much work can I process in this DB2...how many threads can I have?
- My data sharing group has two Internal Coupling Facilities (ICFs).... is this a problem?
- How can I identify runaway jobs before they disrupt my environment.... what can I do about them?

<http://www.ibm.com/software/zseries/telecon/25jul>

## 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](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## A

- active log copy placement 73
- ALLOWAUTOALT(YES) 51
- ALTER GBPOOL 51
- AOR 100
- application considerations 35
- application owning region 100
- application portability 36
- applications
  - best practices 36
- ARM 112
- ARM policy 138
- ASSIST 75
- Auto Alter 50
- Auto Call Selection routine 68
- Automatic Restart Manager 112
- availability 2, 90
- availability best practice 60

## B

- Batch work 93
- best practices
  - summary 135
- BSDS 68, 74
- BSDS information 77
- buffer coherency control 18

## C

- CALL 97
- capacity 2, 8, 90
- castout 22
- castout owner 22
- catalog alias 71
- CDS 67
- central processor complex 2
- CF ix, 11, 19, 39–40, 44, 47, 49, 106, 111, 123–127
- CF commands 115
- CF failure 111
- CFCC 57
- CFRM 25
- CFRM policy 60
- CFSIZER 41

- change log inventory 77
- CHKFREQ 76
- CICS 12, 67, 92
- CICSplex Systems Manager 93, 100
- CLOSE NO 122
- CLOSE YES 122
- CLSN 128
- Command prefix 71
- Commit 36
- component failure 109
- concurrency 37
- concurrency control 18
- configuration flexibility 9, 31
- configurations for growth 32
- Consolidated Service Test 120, 141
- consolidated service test 120
- continuous availability 25
- COORDNTR 75
- copypools 71
- coupling facility 13, 40, 44–45, 66
  - best practices 63
  - GBP sizing 40
  - lock structure sizing 46
  - SCA structure sizing 47
  - sizing strategy 40
- coupling facility control code (CFCC) 57
- coupling facility resource management 25
- CPC 2
- cross invalidation 23, 47
- cross-system extended services 19
- cross-system restart 30
- CTHREAD 75

## D

- data availability 6
- data elements 22
- data set name limit 68
- data sharing ix–x, 3, 7, 15, 18–19, 40, 42, 66–67, 89, 92, 94, 105–106, 109, 121–123
  - architecture 15
  - best performance 123
  - business value 5
  - cost 18

- enabling group 85
- higher transaction rate 10
- introduction 1
- remove a member from the group 88
- roadmap 11
- why implement? 2
- data sharing environment 4
- data sharing group 3–4
- data sharing overhead 123
- DB2 commands 117
- DB2 dump 118
- DB2 location name 69
- DB2 subsystem failure 110
- DDF 12, 69, 92–93
- decision support 33
- design for scalability 18
- directory entries 22, 48
- DIS GBPOOL 52, 117–118
- DIS GROUP 87–88, 117
- disaster recovery 119
- disaster recovery site 88
- distributed data facility 92
- distributed relational database architecture 95
- DRDA port 70
- DSHARE 75
- DSMAX 131
- DSNB250E 107
- DSNDB01 109
- DSNDB07 71
- DSNHDECP 67
- DSNR020I 88
- DSNTIJIN 85
- DSNTIJMV 85
- DSNTIJTM 87
- DSNTIJUZ 85
- DSNZPARM 68, 107, 109, 122, 129, 131
  - TIMESTAMP=YES 68
- DSNZPARMs 74
- DVIPA 91, 94
- dynamic virtual IP addressing 91
- Dynamic workload balancing
  - best practices 103

## E

- ERRSYSMOD 120
- Explicit hierarchical locking 20
- external CF 59

## F

- failures
  - best practices 113
- false contention 124
- FULLTHRESHOLD 50

## G

- GBP 21
  - monitoring 52
- GBP checkpoint 53–54
- GBP optimizations 25
- GBP-dependent 21
- GBP-dependent objects 128
- GCLSN 128
- GDETAIL 52
- generic LU name 70
- global locking 18–19
- GRECP 54, 75, 106–108
- GRECP/LPL recovery
  - best practices 109
- group attach 67
- group buffer pools 21
- group name 66–68
  - restrictions 66
- group restart 110
- group wide outage 58
- growth 2
- GRPNAME 75

## H

- hash 19
- hash class 19
- hashing algorithm 19

## I

- IBM Tivoli Workload Scheduler 98
- ICF 57
- IEFSSNxx 67
- II10817 133
- Implementation tasks 13
- IMS Transaction Manager 93, 101
- incremental growth 7
- indoubt units of recovery (URs) 113
- indoubt URs 113
- inter-DB2 read/write interest 18, 21
- internal coupling facility (ICF) 57
- internal resource lock manager 19



IRLM 19–20, 46–47, 68, 72, 74, 77, 117, 126  
IRLM commands 117  
IRLM group name 72  
IRLM parameter 76  
IRLM subsystem name 72  
ISOLATETIME 112

## L

LIGHT(YES) 113  
list prefetch 108  
L-locks 20, 124  
Load module for subsystem parameters 71  
Location aliases 69  
lock avoidance 36–37, 128  
lock contention 20  
lock structure 19  
lock table entries 46  
locking optimization 20  
LOCKTAB 77  
Log 106  
log data set name 68  
log record sequence number 106  
LOGAPSTG 75  
logging 73  
    best practices 74  
Logical 107  
Logical locks 20  
logical page list 49  
LPARs 26  
LPL 47, 49, 54, 75, 107–110, 137  
LPL recovery 75, 107  
LRDRTHLD 76, 130  
LSRN 106  
LTE 47  
LU name 70  
luname 81

## M

maintenance best practices 120  
MAXDBAT 75  
MAXUSRS 77  
member 3  
Member domain name 71  
MEMBNAME 75  
Monitoring the GBPs 52

## N

naming convention 66  
    best practices 73  
    examples 72  
network requester 94  
Normal restart 110  
number of CF engines 126  
number of threads 131

## O

outage 6

## P

page set 21, 106, 113, 122, 128  
parallel database architectures 16  
Parallel Sysplex 2  
Parallel Sysplex domain name 70  
PCLOSEN 76  
PCLOSET 76, 122  
Physical locks, 20  
PK31073 133  
Planned outages 26  
Planning task 11  
P-locks 20–21, 113  
pooled connections 100  
POPULATECF 28  
Post-implementation tasks 14  
processing capacity 7  
PTF in error 120

## Q

QUIESCE 81, 88

## R

Read For Castout Multiple 126  
REALLOCATE 27  
recommended service upgrade 120  
record list entries 46  
RECOVER 75, 106, 109, 113, 122  
recovery of DB2 objects 106  
Redbooks Web site 142  
    Contact us xiii  
rename existing non-data sharing member 78  
RESTART(LIGHT) 112  
RESYNC port 70  
Retained locks 110, 124  
RETLWAIT 75

RFCOM 126  
RLE 46  
RMF 47, 59, 76, 125–126  
rolling maintenance 26  
routing to multiple DB2s 130  
RRSAF 67  
runaway threads 128

## **S**

SCA 28, 47, 50, 55, 57, 110–111, 113, 128  
scheduled outage 119  
Server Time Protocol 2  
SFM 112  
shared communications area 28  
Shared data 17  
Shared disks 17  
Shared nothing 16  
SLIP 119  
SSID 68, 75  
START DATABASE 107–109  
Stored procedures 97  
STRUCTURE 45  
structure duplexing 53  
STRUCTURE NAME 45  
subsystem identifier 68  
SYNCVAL 76  
SYSLGRNX 122  
Sysplex 2  
Sysplex Distributor 95  
sysplex failure management 112  
System-managed duplexing 53, 55  
systems management 2, 31

## **T**

TCP/IP 95, 131  
thrashing 49  
Tivoli Workload Scheduler for z/OS 98  
transaction manager layer 91  
transaction rates 10  
trickle 53

## **U**

Unplanned outages 29  
URCHKTH 76, 130  
URLGWTH 76, 130  
User-managed duplexing 53

## **V**

VIPA 94  
VIPABACKUP 94  
VIPADEFINE 94  
VIPADYNAMIC 94  
virtual IP address 94  
virtual storage constraint 131

## **W**

WARM 126  
WebSphere 93  
WebSphere Application Server 99  
WLM panel 97  
work file database 71  
workload balancing 2, 7  
    objectives 90  
Workload Manager 92  
Workload spikes 90  
WRKDBP1 71

## **X**

XES 19, 46, 50, 53, 60, 62, 124, 126

## **Z**

z/OS system failure 111, 124  
z890 59  
z9 BC 59



**DB2 for z/OS: Data Sharing in a Nutshell**

(0.2"spine)  
0.17"->0.473"  
90->249 pages







**Redbooks**

# DB2 for z/OS: Data Sharing in a Nutshell

**Get a quick start with  
your DB2 for z/OS  
data sharing  
installation**

**Understand  
alternatives and  
priorities**

**Find support and  
references**

DB2 for z/OS takes advantage of data sharing technology in a z Parallel Sysplex to provide applications with full concurrent read and write access to shared DB2 data. Data sharing allows users on multiple DB2 subsystems, members of a data sharing group, to share a single copy of the DB2 catalog, directory, and user data sets.

Data sharing provides many improvements to availability and capacity without impacting existing applications. The road to data sharing might seem arduous if you are a novice user, but once you have started to learn terminology and gain basic understanding, things will become much easier.

This IBM Redbook is meant to facilitate your journey towards data sharing by providing a cookbook approach to the main tasks in enabling data sharing and workload balancing. Our book does not have all the answers, because it is a brief summary of a large field of knowledge, but it contains the key elements and it points you in the right direction to get more details. Throughout this document we assume that your sysplex environment is set up and a DB2 subsystem exists at a currently supported level.

## **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](http://ibm.com/redbooks)