DB2 for z/OS and OS/390 Version 7
Selected Performance Topics

Learn about bandwidth and response time enhancements when using FICON

Read hints and tips for well-behaved Java applications

Know the performance improvements with CICS TS

Paolo Bruni
Florence Dubois
Claudio Nacamatsu

ibm.com/redbooks
Note: Before using this information and the product it supports, read the information in “Notices” on page xiii.
# Contents

Figures ........................................................................ vii
Tables .......................................................................... ix
Examples ....................................................................... xi
Notices ......................................................................... xiii
Trademarks .................................................................... xiv
Preface ......................................................................... xv
The team that wrote this redbook .................................. xv
Become a published author ........................................... xvi
Comments welcome ..................................................... xvii

## Chapter 1. Introduction .............................................. 1
1.1 Evolution of DB2 V7 .................................................. 2
1.2 Contents of this redbook ............................................ 2

## Chapter 2. FICON .................................................... 5
2.1 Description .............................................................. 6
2.2 Performance measurement environment ..................... 6
2.3 Performance measurement results .............................. 8
   2.3.1 DB2 transaction workload ................................... 8
   2.3.2 DB2 logging ......................................................... 9
   2.3.3 DB2 queries .......................................................... 10
   2.3.4 DB2 utilities ........................................................ 13
   2.3.5 Summary .............................................................. 16
2.4 VSAM striping .......................................................... 16

## Chapter 3. Enhanced Instrumentation Facility .............. 17
3.1 DBM1 storage monitoring ......................................... 18
   3.1.1 z/Architecture and DB2 for z/OS ................................. 18
   3.1.2 Storage manager pool statistics .................................. 19
   3.1.3 DB2 PM statistics report: DBM1 storage statistics ........ 20
   3.1.4 DB2 PM record trace: storage manager pool summary .. 26
3.2 Other new IFCIDs .................................................... 26
   3.2.1 IFCID 0234 ............................................................ 27
   3.2.2 IFCID 0334 ............................................................ 27
3.3 Changed IFCIDs ....................................................... 27
3.4 DB2 PM workstation interface enhancements ............ 29
   3.4.1 New Workstation Online Monitor functions ............... 29
   3.4.2 The Performance Warehouse function ...................... 29

## Chapter 4. Java support .............................................. 31
4.1 JDBC and SQLJ ....................................................... 32
   4.1.1 JDBC overview .................................................... 32
   4.1.2 SQLJ overview ..................................................... 34
4.2 SQLJ versus JDBC .................................................. 35
   4.2.1 Reasons to use SQLJ .............................................. 35
   4.2.2 Reasons to use JDBC ............................................. 37
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.2  DB2 for OS/390 V6 APARs</td>
<td>110</td>
</tr>
<tr>
<td>A.3  DB2 for OS/390 V7 APARs</td>
<td>112</td>
</tr>
<tr>
<td>A.4  DB2 PM V7</td>
<td>115</td>
</tr>
<tr>
<td>A.5  SDK</td>
<td>116</td>
</tr>
<tr>
<td>A.6  OS/390 and z/OS APARs</td>
<td>116</td>
</tr>
<tr>
<td>Abbreviations and acronyms</td>
<td>117</td>
</tr>
<tr>
<td>Related publications</td>
<td>119</td>
</tr>
<tr>
<td>IBM Redbooks</td>
<td>119</td>
</tr>
<tr>
<td>Other resources</td>
<td>119</td>
</tr>
<tr>
<td>Referenced Web sites</td>
<td>121</td>
</tr>
<tr>
<td>How to get IBM Redbooks</td>
<td>121</td>
</tr>
<tr>
<td>IBM Redbooks collections</td>
<td>121</td>
</tr>
<tr>
<td>Index</td>
<td>123</td>
</tr>
</tbody>
</table>
Figures

2-1 Alternative FICON channel/ESS configurations ........................................... 7
2-2 Impact of FICON on Class 2 transaction response time ............................... 8
2-3 Impact of FICON on table space I/O response time ..................................... 9
2-4 Impact of FICON on DB2 log throughput .................................................. 9
2-5 Impact of FICON on non-parallel table scans ........................................... 10
2-6 Impact of FICON on the prefetch I/O response time .................................. 11
2-7 Impact of FICON on parallel table scans .................................................. 11
2-8 Impact of FICON on highly parallel table scans, using one ESS .................... 12
2-9 Impact of FICON on highly parallel table scans, using two ESSs ................. 13
2-10 Impact of FICON on the COPY utility ..................................................... 14
2-11 Impact of FICON on the RECOVER utility ............................................. 14
2-12 Impact of FICON on the LOAD utility ..................................................... 15
2-13 Impact of FICON on the REORG utility ................................................... 16
3-1 STORAGE STATISTICS report layout - IRWW in non data-sharing ............... 21
3-2 STORAGE STATISTICS report layout - IRWW in data-sharing ...................... 24
3-3 STORAGE STATISTICS report layout - Single parallel query ....................... 25
3-4 Storage manager pool summary - IRWW in data-sharing ............................. 26
3-5 Page P-lock counters .............................................................................. 28
3-6 Global contention ..................................................................................... 28
4-1 JDBC ........................................................................................................ 32
4-2 Types of JDBC Drivers ............................................................................. 33
4-3 SQLJ program preparation process ............................................................ 35
4-4 Dynamic versus Static SQL at execution time ........................................... 36
4-5 SQLJ versus JDBC performance ............................................................... 37
4-6 Relative cost of getxxx() methods ............................................................ 41
4-7 CPU overhead: getString() compared to matching getxxx() method .......... 42
4-8 Different throughput using explicit and default context ............................... 45
4-9 Heap size study ....................................................................................... 48
4-10 DB2 PM accounting report: class1 includes JDBC/SQLJ processing .......... 50
4-11 DB2 PM accounting long report: dynamic statement cache ...................... 50
4-12 Sample invocation and output of hprof .................................................... 51
4-13 SQLJ/JDBC trace ................................................................................... 52
4-14 Tests environment ................................................................................... 53
4-15 Normalized transactions per second ........................................................ 53
4-16 SQLJ/JDBC performance study ............................................................... 56
4-17 Distribution of the processing cost in each measurement ......................... 57
5-1 CICS/DB2 attachment architecture ............................................................ 62
5-2 DB2CONN screen fragment - TCBLIMIT .................................................. 63
5-3 Threads and TCBs .................................................................................... 64
5-4 Threads and TCBs - THREADLIMIT ......................................................... 65
5-5 -DISPLAY THREAD ............................................................................... 67
5-6 CEMT INQUIRE TASK outputs .............................................................. 68
5-7 CICS system dump .................................................................................. 68
5-8 DSNC DISP STAT output ........................................................................ 69
5-9 DB2 PM report - thread reuse information ............................................... 70
5-10 Sample DFHSTUP JCL ......................................................................... 71
5-11 DFHSTUP output: Resource Statistics - Request ................................... 71
5-12 DFHSTUP output - Resource Statistics - Performance ............................ 72

© Copyright IBM Corp. 2002. All rights reserved.
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5-13</td>
<td>CPU accounting</td>
<td>72</td>
</tr>
<tr>
<td>5-14</td>
<td>User response time</td>
<td>74</td>
</tr>
<tr>
<td>5-15</td>
<td>DB2ENTRY screen fragment</td>
<td>74</td>
</tr>
<tr>
<td>5-16</td>
<td>Thread protocol sequence</td>
<td>76</td>
</tr>
<tr>
<td>5-17</td>
<td>Resign-on</td>
<td>77</td>
</tr>
<tr>
<td>5-18</td>
<td>Queued at create thread</td>
<td>78</td>
</tr>
<tr>
<td>5-19</td>
<td>Pre-OTE and OTE CICS/DB2 attach</td>
<td>81</td>
</tr>
<tr>
<td>5-20</td>
<td>Cost savings with OTE</td>
<td>83</td>
</tr>
<tr>
<td>6-1</td>
<td>When real time statistics are updated</td>
<td>92</td>
</tr>
<tr>
<td>7-1</td>
<td>DB2 OLAP Server for OS/390 V1.1 versus V7.1 using different I/O subsystems</td>
<td>100</td>
</tr>
<tr>
<td>7-2</td>
<td>DB2 OLAP Server for OS/390 V1.1 versus V7.1 using different type of hardware</td>
<td>101</td>
</tr>
<tr>
<td>7-3</td>
<td>Average calculation time of the cubes</td>
<td>103</td>
</tr>
<tr>
<td>7-4</td>
<td>CPU utilization during the calculation of the cubes</td>
<td>103</td>
</tr>
<tr>
<td>7-5</td>
<td>Average load time of the cubes</td>
<td>105</td>
</tr>
<tr>
<td>7-6</td>
<td>CPU utilization during the load of the cubes</td>
<td>105</td>
</tr>
<tr>
<td>7-7</td>
<td>Average calculation time: 64-bit versus 31-bit</td>
<td>106</td>
</tr>
</tbody>
</table>
Tables

3-1 Virtual storage consumers in a “typical” DB2 installation .................................................. 20
3-2 Example 1: Virtual storage consumers ................................................................. 23
3-3 Example 2: Virtual storage consumers ................................................................. 24
3-4 Example 3: Virtual storage consumers ................................................................. 25
3-5 New IFCIDs ................................................................................................................. 26
4-1 JDBC/SQLJ Enhancements by APARs ................................................................. 39
4-2 Mapping DB2 data types to Java data types ..................................................... 41
4-3 Improvement in Java column processing cost - getxxx() methods ............... 55
4-4 improvements in Java column processing cost - setxxx() methods ............... 55
6-1 RTS objects .................................................................................................................. 89
6-2 When real time statistics are externalized .......................................................... 91
6-3 Updating real time statistics .................................................................................... 92
6-4 Real time statistics and DB2 utilities ................................................................. 93
7-1 Elapsed times for single-threaded load and calculation ........................................ 102
A-1 DB2 V5 performance related APARs ................................................................. 110
A-2 DB2 V6 performance related APARs ................................................................. 110
A-3 DB2 V7 performance related APARs IRLM ................................................... 113
A-4 IRLM 2.1 APARs ....................................................................................................... 115
A-5 DB2 PM V7 APARs ................................................................................................. 115
A-6 SDK APARs .............................................................................................................. 116
A-7 OS/390 DB2 related APARs .................................................................................... 116
Examples

3-1 How to turn on Statistics Class 6 trace .................................................. 19
3-2 RETRACE command ................................................................. 26
4-1 Simple JDBC application .......................................................... 32
4-2 Same program using JDBC and SQLJ ........................................ 34
4-3 DataSource definition ............................................................... 43
4-4 Connection pooling ................................................................. 43
4-5 Explicit connection context ........................................................ 45
4-6 Named iterator ........................................................................... 46
4-7 Positioned Iterator .................................................................... 46
Notices

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

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

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

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

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

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

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

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

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

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

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

- AIX®
- CICS®
- CICS/ESA®
- CICSplex®
- DataJoiner®
- DataPropagator™
- DB2®
- DB2 Connect™
- DB2 OLAP Server™
- DFS™
- DRDA®
- Enterprise Storage Server™
- ESCON®
- FICON™
- IBM®
- IBM eServer™
- IMS™
- MVS™
- OS/390®
- PerformTM
- QMF™
- RACF®
- RAMAC®
- RedbooksTM
- Redbooks(logo)TM
- RETAIN®
- RMF™
- S/390®
- SP™
- System/390®
- VisualAge®
- VTAM®
- WebSphere®
- z/Architecture™
- z/OS™
- zSeries™

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

- Lotus®
- Word Pro®

The following terms are trademarks of other companies:

- ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

- C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.

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

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

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

Performance measurements are ongoing during the life of each release of DB2 for OS/390. DB2 for z/OS and OS/390 Version 7 (DB2 V7 throughout this document) has introduced several enhancements in the areas of performance and availability, and other enhancements are currently being added.

Most of these enhancements and the related performance measurements implemented in the Silicon Valley Laboratory have been documented in the IBM Redbook DB2 for z/OS and OS/390 Version 7 Performance Topics, SG24-6129.

However, information on new DB2 functions and their synergy with the evolving zSeries platform is of great value for strategic investments, and more performance measurements are under way. Currently these measurements have included topics such as FICON, Java support, CICS interface, and DB2 OLAP Server.

This IBM Redbook is meant to provide an update on the new measurements that have been implemented, and to point out the performance related maintenance that has been shipped after general availability of DB2 for z/OS Version 7. The information here contained is intended to help managers and professionals understand and evaluate the applicability to their environment of these recent functions of DB2 V7.

This IBM Redbook replaces the IBM Redpaper that was made available in February 2002, with the same title, and provides more up-to-date information.

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, San Jose Center.

Paolo Bruni is a Certified Consultant IT Architect on assignment at the International Technical Support Organization, San Jose Center, where he conducts projects on all areas of Data Management. Before joining the ITSO in 1998, Paolo worked in IBM Italy. During his many years with IBM, in development and in the field, Paolo’s work has been mostly related to database systems.

Florence Dubois is a DB2 for z/OS and OS/390 Specialist at the IBM ATS Products and Solutions Support Center (PSSC), Montpellier, France. For the last six years Florence has been involved in several large performance benchmarks, mainly in the ERP area (SAP R/3 and PeopleSoft). She has also been providing on-site DB2 performance reviews for large EMEA customers. Her areas of expertise include DB2 performance and data sharing.

Claudio Nacamatsu is a DB2 Specialist at the IBM Support Center, in São Paulo, Brazil, where he provides support to customers in the areas of DB2 for OS/390 and IBM replication solution products such as DataJoiner and DataPropagator. Before joining IBM, two years ago, Claudio had been working for seven years as DB2 for OS/390 system programmer and DBA at Petrobras, a large Brazilian oil company. He holds a Bachelor’s degree in Computer Science from São Paulo University - USP.
Thanks to the following people for their contributions to this project:

Yvonne Lyon
Bart Steegmans
Phil Wakelin
International Technical Support Organization, San Jose Center

Jeff Berger
Chuck Bonner
John Campbell
Willie Favero
Gene Fuh
James Guo
Robert Gilles
Akiko Hoshikawa
Gopal Krishnan
Roger Miller
Bill Malloy
Todd Munk
Mai Nguyen
Jim Pickel
Jim Pizor
Dave Raiman
Jim Ruddy
Akira Shibamiya
Nigel Slinger
Jim Teng
Annie Tsang
Yoichi Tsuji
Frank Vitro
Jane Wang
Maryela Weihrauch
Chung Wu
Ron Yorita
Casey Young
IBM Silicon Valley Lab

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'll team with IBM technical professionals, Business Partners and/or customers.

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

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

ibm.com/redbooks/residencies.html
Comments welcome

Your comments are important to us!

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

▶ Use the online Contact us review redbook form found at:
  ibm.com/redbooks

▶ Send your comments in an Internet note to:
  redbook@us.ibm.com

▶ Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. QXXE  Building 80-E2
  650 Harry Road
  San Jose, California 95120-6099
Introduction

Our main objective in this redbook is to bring the reader up-to-date with what has been made available through standard maintenance to DB2 UDB for OS/390 and z/OS Version 7 in terms of performance functions; some measurements that have taken place after the general availability of DB2 V7 in March 2001; and the major considerations that can be drawn from these measurements.

In this chapter, we provide an introduction to the contents of this redbook and summarize the impact of the new performance functions.
1.1 Evolution of DB2 V7

With Version 7, DB2 UDB for z/OS and OS/390 continues to meet customer needs in a rapidly changing business environment. DB2 V7 has delivered most of its enhancements at general availability (GA), and it is delivering much less function through maintenance changes than DB2 V5 or V6 did.

You may remember that the changes introduced to DB2 V5 and V6 after GA deserved a dedicated enhancements redbook for each release (see DB2 Server for OS/390 Version 5 Recent Enhancements - Reference Guide, SG24-5421 and DB2 UDB Server for OS/390 Version 6 Technical Update, SG24-6108.) With DB2 V7, the DB2 development organization has committed to a substantial reduction in the impact of new functions delivered through the service stream.

Part of this commitment results in less function being delivered this way. Yet, DB2 is a very lively product with a large community of diverse users, and enhancements did and do take place through maintenance. Another part of the commitment is that, when more function is delivered through service, then much care is taken in avoiding impacts and problems by improvements in the design, development, and testing processes.

In this redbook we describe the major, performance related, aspects of these enhancements; we report several new measurements; and we present some conclusions and recommendations that might help in increasing the value of your investment in DB2.

1.2 Contents of this redbook

Here are the performance items we consider in this redbook:

- **FICON**
  FICON is the new I/O interface based on the industry standard Fibre Channel architecture. Measurements demonstrate the improvements that FICON channels can provide for various DB2 workloads, and that can result in the following benefits:
  - A better response time is achieved with a quarter as many channels for transaction workloads.
  - The maximum DB2 log throughput can increase by 17%.
  - Non-parallel table scans are twice as fast. Highly parallel table scans can achieve almost equivalent response time with a quarter as many channels. DB2 queries can benefit from this improvement.
  - REORG, COPY and LOAD can run 35% faster. RECOVER can run 56% faster.

- **Enhanced Instrumentation Facility**
  We describe the recent additions to the DB2 Instrumentation Facility, paying special attention to the new IFCIDs that provide detailed information on the way that DB2 uses virtual storage in the DBM1 address space.

- **Java support**
  This is a new important area where recently gained experience, performance measurements, and several enhancements can really help in improving your Java developed applications. We describe the latest enhancements to JDBC and SQLJ native environments, we present some guidelines for high performance using JDBC/SQLJ, we report some measurements on the performance improvements, and we list some hints and tips on analyzing performance.
CICS interface
We summarize the CICS/DB2 enhancements introduced by CICS TS V2.2 and provide some hints and tips about monitoring and tuning the CICS/DB2 attachment facility. We describe the enhancements provided by CICS TS V2.2 specific to the CICS/DB2 attachment. One enhancement is the use of the Open Transaction Environment (OTE); measurements in ideal environment show that this function can provide improvements up to 40% in transaction throughput.

Real time statistics
The new function real time statistics (RTS) is a functional enhancement that has been introduced in DB2 V7 by maintenance after GA. This function provides the statistics that end users, or automated task schedulers, can use to determine which objects require REORG, RUNSTATS or COPY.

DB2 OLAP Server on zSeries
Recent measurements have highlighted two performance topics:
– The first topic is DB2 OLAP Server for OS/390 V1.1 versus V7.1 performance comparison
  Performance measurements at IBM’s Silicon Valley Lab demonstrate that V7.1 performs substantially better than V1.1:
  • Measurements on identical hardware and software configurations for loading and calculations improved 37% to 50%.
  • Measurements comparing hardware available at the time of V1.1 general availability vs. current hardware with V7.1, along with tuning, results in 2.6 times to 7.3 times improvements.
– The second topic covers throughput and scalability characteristics of DB2 OLAP Server for OS/390 V7.1
  On a 12-CPU zSeries 900 system:
  • For a CPU-bound cube, running 12 concurrent calculations required only 9% longer than a single cube. This is an 11 times improvement in throughput, compared with running calculations serially.
  • For an I/O-bound cube, running 20 concurrent calculations required 20% longer than a single cube. This resulted in a 16.9 times improvement in throughput, compared with running calculations serially.

APARs
In Appendix A, “Recent performance and availability maintenance” on page 109 we list recent APARs that are of interest in these areas.
FICON

FICON is the new I/O interface based on the industry standard Fibre Channel Architecture. The new FICON channels offer improvements over the traditional ESCON channels and can provide significant performance benefits to key customer workloads.

In this chapter we describe how the new FICON channels can positively impact DB2 performance in various environments. The measurements are divided into the following categories:

- DB2 transaction workload
- DB2 logging
- DB2 queries
- DB2 utilities
2.1 Description

The new FICON channels support many improvements over the previously available ESCON channels, and relieve many ESCON architectural limitations. These improvements include:

- **Increased data transfer rate**
  
The data transfer rate across the link is limited to 17 MB/sec on an ESCON channel. The FICON link data transfer rate is 100 MB/sec.

- **Increased distance and no droop**
  
  With ESCON channels, the distance from a host processor to the control units was limited to 3 km without switches or repeaters, and up to 43 km with switches or repeaters. Although the data rate dropped rapidly after 9 km, FICON channels now support distances up to 20 km without repeaters, and up to 100 km with repeaters, without data transfer rate droop.

- **Channel aggregation**
  
  Because of the increased transfer rate and the improved protocol, a single FICON channel can replace multiple ESCON channels while providing equal or better performance. This means fewer channels, director ports and control unit ports, and simpler configuration to manage. This allows you to reduce infrastructure costs.

- **Increased addressability**
  
  The current addressing limitation for ESCON is 1,024 unit addresses per channel. FICON channels now support up to 16,384 unit addresses.

The redbook *FICON Native Implementation and Reference Guide*, SG24-6266, covers the planning and implementation of FICON channels, operating in FICON native mode for the IBM zSeries 900 and 9672 Generation 5 (G5) and Generation 6 (G6) processors.

A detailed discussion of the FICON technology is included in the white paper *ESS FICON Channel Attachment* by Phil Mills. It is available at the Web site:


FICON native channels can now be exploited to connect new or existing IBM zSeries 900, S/390 9672 G5, or G6 processors; and IBM Enterprise Storage Servers (ESS) Models F10 or F20.

A set of DB2 measurements were executed at IBM Silicon Valley Laboratory to evaluate the performance benefits on DB2 of FICON channels over the traditional ESCON channels. We describe the results of some of these tests in this chapter. For more details on this evaluation you can ask your IBM representative to consult the white paper *DB2 for OS/390 Performance using FICON Channels* by Jeffrey Berger, James Guo, Frank Vitro, and Ron Yorita available at:

http://w3.ibm.com/sales/systems/disk

Some of the issues associated with using RMF to do capacity planning and performance analysis are also discussed in this white paper.

2.2 Performance measurement environment

All the measurements were performed using one or two ESS Model F20s, with 16 GB of cache and 384 MB of non-volatile storage (NVS). Three Parallel Access Volumes (PAVs) were defined for each 3390-3 logical volume.
In this environment, 16 ESCON channels and 8 FICON channels were attached to each ESS. They were alternatively online – none of the tests were done with a combination of the two channel types.

The FICON channel cables used were 9 micron single mode fiber cables using the long wavelength laser 1300 nano meter transmission, configured in a switched point-to-point connection via an IBM 2042-001 (INRANGE FC/9000-64) FICON director. These measurements do not reflect the recent generation of FICON Adapter Express Card; even better performance could be expected with the currently available new cards.

Most of the measurements were done on a zSeries 900 processor. DB2 queries have been evaluated on both zSeries 900 and G6 processors.

All the measurements were conducted using OS/390 V2R10 and DB2 V7.

Figure 2-1 shows the different FICON channel/ESS configurations that were measured, all using a single host processor.

Diagrams A and B depict 4 FICON channels, while diagram C depicts 8 channels. The difference between diagrams A and B is that diagram B uses twice as many ESS host adapters (HAs), that is 4 versus 8; this can only be done using a FICON director.

Diagrams D, E, and F depict FICON configurations for 2 ESSs using daisy chaining, that is sharing one set of FICON channels. Diagrams D and E depict 4 channels, while diagram F depicts 8 channels. Diagram D illustrates 4 HAs for each ESS, while diagrams E and F represent 8 HAs for each ESS. All three of these configurations contain more HAs than channels, but the configuration in E has twice as many HAs as one would ordinarily expect with two ESSs, using a total of 16 HAs with only 4 channels.
The ESCON channel configurations are not diagrammed. Attached to each ESS were 16 ESCON channels, divided among several independent ESCON directors. The ESCON channels were dedicated to their respective storage server.

2.3 Performance measurement results

In this section, we provide a summary of the results of the performance measurements.

2.3.1 DB2 transaction workload

The performance measurements were executed using the IBM Relational Warehouse Workload (IRWW). The database size was 30 GB, which is about twice the size of the ESS cache. The workload was run on a zSeries 900 processor employing two dedicated engines and achieving 64% CPU busy. The DB2 buffer pool size was less than 500 MB, achieving only a 16% buffer pool hit ratio and 4693 I/Os per second. Such a high I/O rate for a 30 GB database is a very high access density which older IBM control units could not sustain, but it can easily be accommodated by one ESS. Because the ESS cache hit ratio was about 93%, the IRWW was limited more by channel performance than disk performance, which makes it ideal for studying channel performance.

Figure 2-2 shows that FICON reduced the Class 2 (time spent in DB2) transaction response time by 20% compared with ESCON, with 4 to 1 reduction in channels.

Figure 2-3 illustrates that the table space I/O response time improved by 28%, all with a four to one reduction in the number of channels.
2.3.2 DB2 logging

The performance measurements were executed with the specific objective of evaluating the DB2 log throughput with FICON. Logging rates were measured using an intensive sequential insert workload. No channel contention was present in any of the configurations.

Figure 2-4 shows that FICON increased the log throughput by 17% compared with ESCON.
FICON logging rates were also measured when using DFSMS striping for the log. More than 20 MB/sec were achieved with two stripes, and 30 MB/sec were achieved with eight stripes. After this value, as more and more stripes are added, the returned benefits tend to diminish rapidly.

For more information about log striping and how to enable it, see the redbook *DB2 for z/OS and OS/390 Version 7 Performance Topics*, SG24-6129.

### 2.3.3 DB2 queries

**Single channel comparison**
The first series of measurements were done using a single channel, either ESCON or FICON.

**Non-parallel table scan**
Figure 2-5 illustrates the result of DB2 prefetch\(^1\) scanning a non-partitioned table. It shows that FICON doubled the throughput over ESCON.

![Figure 2-5   Impact of FICON on non-parallel table scans](image)

Note that the FICON throughput shown in Figure 2-5 was obtained on a zSeries 900 processor. Figure 2-6 shows a twofold reduction in the prefetch I/O response time.

---

\(^1\) For transactions, DB2 prefetch I/Os usually fetch 32 4 KB pages, that is 131,072 bytes per I/O.
Parallel table scan
Figure 2-7 shows the throughput of a multi-partition scan with a single channel.

Partition scan
Throughput (MB/sec)

Figure 2-7  Impact of FICON on parallel table scans
If we compare the throughput for partitioned case in Figure 2-7 versus the non-partitioned one in Figure 2-6, we can observe that the throughput of ESCON increases only slightly (10%) over a non-partitioned scan (14.6 versus 13.8 MB/sec), but the throughput of FICON almost doubles (51 versus 28 MB/sec). This is due to the fact that, while a single partition can nearly saturate one ESCON channel, it takes two partitions to nearly saturate a FICON channel.

Note that the FICON throughput shown in Figure 2-7 was obtained on a zSeries 900 processor.

**Multiple channels with many partitions**

In this case the performance measurements were executed using a decision-support workload modeled on the TPC-D specification. The database size was 30 GB and had 8 tables. The largest of these tables had 60 partitions. These 60 partitions were first spread across the 16 LCUs of a single ESS, and then spread across 2 ESSs.

Figure 2-8 shows the aggregate throughput when scanning all 60 partitions in parallel, using a single ESS, with 16 ESCON and with FICON configurations A, B, and C, which were shown in Figure 2-1.

With a single ESS, 4 FICON channels were 15% slower than 16 ESCON, but 8 FICON were 19% faster. Although it is best to have 8 FICON channels, the incremental benefit is not double that of 4 FICON because the ESS becomes saturated with 8 FICON with this query workload. In the intermediate case, when 4 FICON channels were coupled with 8 ESS HAs, FICON was only 8% slower than 16 ESCON. Given 4 FICON channels, the extra HAs boosted the throughput by 10% (192 versus 175 MB/sec).

Figure 2-9 shows the performance of the same query as Figure 2-8, except that the partitions were spread across two ESSs. Each ESS had its own set of 16 ESCON channels, but the performance were first measured with only 8 ESCON channels on each ESS (for a total of 16), and then all 32 ESCON channels. For the FICON measurements, configurations D and F shown in Figure 2-1 were used.
The two ESSs were independent of each other. They were using different channels and ESCON directors. Therefore, the throughput with two ESSs and 32 ESCON channels was double that of one ESS and 16 ESCON (412 versus the 206 MB/sec in Figure 2-8 on page 12). In all of these ESCON cases, the channel utilizations were close to 99%, indicating that additional storage servers could not provide higher throughput if they shared the same channels.

As was the case with one ESS, a four to one reduction in the number of channels with two ESSs results in only a 10% reduction in the throughput (372 versus 412 MB/sec), again with twice as many FICON HAs as FICON channels. Furthermore, the increase in the throughput gained by adding a second ESS sharing the same set of 8 FICON channels increased by 52% (372 versus 245 MB/sec in Figure 2-8 on page 12), compared to only the 5% increase observed with 16 ESCON channels (217 versus 206 MB/sec in Figure 2-8 on page 12). This proves that the 8 FICON channels were not saturated when serving one ESS, but the 16 ESCON channels were nearly so.

### 2.3.4 DB2 utilities

For the DB2 utilities, the ESCON measurements were done with all 16 ESCON channels online, and the FICON measurements were done with all 8 FICON channels online. However, neither the ESCON nor FICON channels were constrained.

**COPY utility**

The COPY utility was first measured with FICON channels without striping. Then two DFSMS stripes were defined for the image copy output data set in order to demonstrate how striping can be used to further improve performance.

Figure 2-10 shows that with 8 FICON channels, the COPY utility provides 34% more throughput than with 16 ESCON channels.

An additional 36% improvement can be obtained with striping.
Figure 2-10  Impact of FICON on the COPY utility

RECOVER utility
Figure 2-11 shows that with 8 FICON channels, the RECOVER utility provides 56% more throughput than with 16 ESCON channels.
LOAD utility
The LOAD utility was measured using a 1.2 GB table (10 million rows, 117 byte rows, and 26 columns per row). One index containing two columns was defined. Figure 2-12 shows that each of the LOAD utility phases improves the elapsed time and, overall, the LOAD utility runs 31% faster.

![LOAD utility with index](image)

*Figure 2-12 Impact of FICON on the LOAD utility*

The CPU time for ESCON and FICON is the same (111 sec). The I/O time is still higher than the CPU time on zSeries 900, but the ratio of CPU time to elapsed time is reduced with FICON. With many partitions, it is likely that the LOAD utility will become CPU bound.

REORG utility
The elapsed time of the REORG utility (with SORTDATA specified) was measured using the same table as described for the LOAD utility, with a 50% cluster ratio.

REORG was also measured with two DFSMS stripes for the SYSREC data set, to demonstrate how striping can be used to further reduce the elapsed time.

Again, FICON improved all phases of REORG elapsed time. Figure 2-13 shows that the REORG utility runs 35% faster with FICON. An additional 14% reduction can be obtained with striping.
With FICON and 2 SYSREC stripes, the ratio of zSeries 900 CPU time to elapsed time was nearly 50% (91 sec for the CPU time versus 188 sec for the elapsed time).

2.3.5 Summary

The measurements reported in this chapter demonstrate the improvements that FICON channels can provide for various DB2 workloads when comparing to ESCON channels. These improvements can result in the following benefits:

- **DB2 transaction workload**
  A better response time is achieved with a quarter as many channels.

- **DB2 logging**
  The maximum DB2 log throughput, without striping, is 17% higher. With striping the throughput reaches 30 MB/sec.

- **DB2 queries**
  Non-parallel table scans are twice as fast. Highly parallel table scans can achieve almost equivalent response time with a quarter as many channels.

- **DB2 utilities**
  REORG, COPY, LOAD and RECOVER show significant improvements in throughput and elapsed time.

2.4 VSAM striping

Current DB2 usage of VSAM striping is mentioned in the redbook *DB2 for z/OS and OS/390 Version 7 Performance Topics*, SG24-6129. A restriction has been removed by APAR PQ54580 (see A.3, “DB2 for OS/390 V7 APARs” on page 112) and more measurements for more variety of data are still under way.
Enhanced Instrumentation Facility

In this chapter we describe the recent additions to the DB2 Instrumentation Facility, paying special attention to the new IFCIDs that provide detailed information on the way that DB2 uses virtual storage in the DBM1 address space.
3.1 DBM1 storage monitoring

Each MVS address space has an implicit limit of 2 GB based on the 31-bit addressing. This corresponds to a limit of 16 MB below the line, and 2032 MB above the line. DB2 manages storage into several subpools by issuing GETMAIN and FREEMAIN.

The biggest consumer of virtual storage is the DB2 DBM1 address space where storage is allocated for the DB2 subsystem and for each individual thread. The major contributors to the storage demand above the line are:

- The Virtual Buffer Pool
- The EDM pool
- The user threads
- The system threads
- Compression dictionaries
- Local dynamic statement cache

A growing number of DB2 for OS/390 installations are running near the 2 GB limit for the DBM1 address space and many will be pushing this limit in the near future. This is due to the fast growth and increased spiking brought on by new types of workloads like e-business applications or enterprise applications (PeopleSoft, SAP, Siebel, etc.).

Short-on-storage conditions can be received at GETMAIN time and a critical DB2 task may fail. The RMF Virtual Storage Private Area Report identifies potential storage shortages and trends of storage consumption. An amount of storage still available above the line of less than 200 MB requires some action. A longer term solution will be provided by the full support of the new z/Architecture by DB2 (see 3.1.1, “z/Architecture and DB2 for z/OS” on page 18.)

In the meantime, close monitoring and tuning is necessary. More data in storage monitoring is provided by current enhancements to DB2’s Instrumentation Facility that will produce new records edited by the corresponding enhancements to DB2 PM. This is a work in progress, with the formats not completely finalized. For recent information on maintenance, it is recommended to monitor the DB2 storage INFO APARs II04309 and II10817.

3.1.1 z/Architecture and DB2 for z/OS

In z/OS V1.2, IBM delivered the initial basic 64-bit virtual storage management support. This is a major milestone of the new z/OS 64-bit operating environment. With this basic 64-bit virtual storage support, an application address space can have $2^{**}64$ virtual addresses with backing by real storage as needed. Over the next two years, z/OS will deliver the rest of the 64-bit virtual storage functions to meet the needs of growing e-business application environments.

The objective of the white paper IBM eServer zSeries 900 z/OS 64-bit Virtual Storage Roadmap, GM13-0076, is to describe for software vendors and customers how IBM will provide 64-bit virtual storage support through a multi-stage plan. The ultimate goal of this plan is to provide a hardware and software platform that functions as the execution environment of choice for the e-business workloads that will dominate future commercial data processing while maintaining today's important applications. IBM hopes software vendors and customers will use this white paper to help them make the right business decisions as they upgrade existing applications and develop new e-business applications that need large virtual storage.
IBM plans to deliver 64-bit virtual storage addressing for the DB2 for z/OS product in the future release. The future release of DB2 for z/OS, with 64-bit virtual address support, can only execute on IBM zSeries 900 (z900), or equivalent, running z/OS V1R3, or later. DB2 V6 (5645-DB2) and V7 (5675-DB2) already support 64-bit real storage addressing for data space buffers.

3.1.2 Storage manager pool statistics

Two new IFCIDs, 0225, and 0217, have been added in DB2 V7 to record DBM1 storage usage statistics. They give you the ability to more effectively monitor DBM1 address space so that actions can be taken to alleviate or avoid storage shortage conditions. They are supported by the corresponding enhancements to DB2 PM reporting.

**IFCID 0225**

IFCID 0225 provides you with summary information on the storage usage in the DBM1 address space (while IFCID 0217 provides the detailed information). This IFCID is contained in Statistics Class 6 and is recorded at the DB2 statistics interval.

Statistics Class 6 trace can be turned on with the START TRACE command or the MODIFY TRACE command (if the Statistics trace is already started). An example of these commands is listed in Example 3-1. The addition of this class to the statistics should not add noticeable overhead to the trace.

Example 3-1   How to turn on Statistics Class 6 trace

```
-STA TRACE(S) CLASS(1,3,4,5,6)
-MOD TRACE(S) TNO(1) CLASS(1,3,4,5,6)
```

**IFCID 0217**

IFCID 0217 provides you with detailed information on the storage usage in the DBM1 address space. This IFCID is contained in Global Class 10 and is recorded at the DB2 statistics interval. It is mainly a serviceability aid.

This record details the amount of available storage in the DBM1 address space, the amount of storage for MVS use, the total GETMAINed stack storage, and the total getmained storage. This is followed by information on each DBM1 storage pool, and each agent storage pool. If there are more than 250 such pool entries, they will overflow to another IFCID 0217 record. For each pool, the total storage used is recorded. For agent pools, the thread is identified by authorization ID, correlation ID, connection name, and plan name.

**DB2 PM support**

IFCID 0225 is supported in DB2 PM V7 statistics report if the PTF UQ56531 for APAR PQ50902 is applied and the records were created by a DB2 V7. This includes support of the performance database (using the FILE option) as well as upgrade (ALTER TABLE...) of already installed performance table DB2PM_STAT_GENERAL. IFCID 0217 and 0225 are supported by the DB2 PM V7 record trace, independently from which DB2 version was used when they were created.
Virtual storage budget
In the article *DB2 UDB for OS/390 Storage Management*, IDUG Solutions Journal, Spring 2000, available from http://www.idug.org/member/journal/mar00/storage.cfm, the authors explain a methodology on how to estimate your virtual storage budget and provide relief to virtual storage constraints inside the DBM1 address space. You can now use the storage statistics report to easily identify the storage consumption of each component as shown in the following sections.

### 3.1.3 DB2 PM statistics report: DBM1 storage statistics

The components that are responsible for the major part of the virtual memory consumption in the DBM1 address space are now reported in section DBM1 STORAGE STATISTICS of the DB2 PM statistics long report.

**Who uses virtual storage in DBM1 address space?**
Table 3-1 illustrates the major virtual storage consumers in a *typical* DB2 installation.

<table>
<thead>
<tr>
<th>Consumers of virtual storage</th>
<th>“Typical” customer usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual buffer pools</td>
<td>40 to 800 MB</td>
</tr>
<tr>
<td>EDM pool</td>
<td>20 to 400 MB</td>
</tr>
<tr>
<td>User thread storage</td>
<td>0.1 to 2 MB per active user thread</td>
</tr>
<tr>
<td>System thread storage</td>
<td>50 KB per system thread</td>
</tr>
<tr>
<td>Compression dictionary</td>
<td>up to 64 KB for each open compressed data set</td>
</tr>
<tr>
<td>Local dynamic statement cache</td>
<td>0 to 300 MB</td>
</tr>
<tr>
<td>Others</td>
<td>200 to 400 MB</td>
</tr>
</tbody>
</table>

In this section, we discuss sample DB2 PM statistics reports corresponding to the following cases:
- Example 1: Transaction workload in a non-data sharing environment
- Example 2: Transaction workload in a data sharing environment
- Example 3: Single parallel query
Example 1: Transaction workload in a non-data sharing environment

The IRWW was run in a non-data sharing environment. Figure 3-1 is an extract of the DB2 PM statistics long report. It corresponds to the DBM1 STORAGE STATISTICS section.

<table>
<thead>
<tr>
<th>DBM1 STORAGE STATISTICS (MB)</th>
<th>QUANTITY</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOTAL GETMAINED STORAGE</td>
<td>440.17 (B)</td>
</tr>
<tr>
<td>VIRTUAL BUFFER POOLS</td>
<td>407.71 (F)</td>
</tr>
<tr>
<td>EDM POOL</td>
<td>19.53 (K)</td>
</tr>
<tr>
<td>COMPRESSION DICTIONARY</td>
<td>0.00 (G)</td>
</tr>
<tr>
<td>CASTOUT BUFFERS</td>
<td>0.00</td>
</tr>
<tr>
<td>DATASPACE LOOKASIDE BUFFER</td>
<td>0.00</td>
</tr>
<tr>
<td>TOTAL VARIABLE STORAGE</td>
<td>40.99 (C)</td>
</tr>
<tr>
<td>TOTAL AGENT SYSTEM STORAGE</td>
<td>17.43 (V)</td>
</tr>
<tr>
<td>TOTAL AGENT LOCAL STORAGE</td>
<td>26.64 (P)</td>
</tr>
<tr>
<td>RDS OP POOL</td>
<td>3.32 (Q)</td>
</tr>
<tr>
<td>RID POOL</td>
<td>0.57 (L)</td>
</tr>
<tr>
<td>PIPE MANAGER SUB POOL</td>
<td>0.00</td>
</tr>
<tr>
<td>LOCAL DYNAMIC STMT CACHE CTL BLKS</td>
<td>0.99 (M)</td>
</tr>
<tr>
<td>LOCAL DYNAMIC STMT CACHE STMT POOL</td>
<td>0.00 (N)</td>
</tr>
<tr>
<td>BUFFER &amp; DATA MANAGER TRACE TBL</td>
<td>6.59 (R)</td>
</tr>
<tr>
<td>VIRTUAL POOL CONTROL BLOCKS</td>
<td>12.74 (G)</td>
</tr>
<tr>
<td>HIPERPOOL CONTROL BLOCKS</td>
<td>0.00 (H)</td>
</tr>
<tr>
<td>DATASPACE BP CONTROL BLOCKS</td>
<td>0.00 (I)</td>
</tr>
<tr>
<td>TOTAL FIXED STORAGE</td>
<td>0.09 (D)</td>
</tr>
<tr>
<td>TOTAL GETMAINED STACK STORAGE</td>
<td>7.78 (E)</td>
</tr>
<tr>
<td>STORAGE CUSHION</td>
<td>91.85 (U)</td>
</tr>
<tr>
<td>TOTAL DBM1 STORAGE</td>
<td>489.03 (A)</td>
</tr>
<tr>
<td>TOTAL NUMBER OF ACTIVE USER THREADS</td>
<td>75.00 (T)</td>
</tr>
<tr>
<td>TOTAL STORAGE FOR ALL THREADS</td>
<td>44.33 (S)</td>
</tr>
<tr>
<td>NUMBER OF PREFETCH ENGINES</td>
<td>39.00</td>
</tr>
<tr>
<td>NUMBER OF DEFERRED WRITE ENGINES</td>
<td>300.00</td>
</tr>
<tr>
<td>NUMBER OF CASTOUT ENGINES</td>
<td>0.00</td>
</tr>
<tr>
<td>NUMBER OF GBP WRITE ENGINES</td>
<td>0.00</td>
</tr>
<tr>
<td>NUMBER OF P-LOCK/NOTIFY EXIT ENGINES</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 3-1 STORAGE STATISTICS report layout - IRWW in non data-sharing

In this example, the TOTAL DBM1 STORAGE (A) is 489.03 MB.

This total can be split up into the following components:

- TOTAL GETMAINED STORAGE (B) = 440.17 MB
- TOTAL VARIABLE STORAGE (C) = 40.99 MB
- TOTAL FIXED STORAGE (D) = 0.09 MB
- TOTAL GETMAINED STACK STORAGE (E) = 7.78 MB

Virtual buffer pools are usually the major consumers of DB1 storage.

- The storage required for the virtual buffer pools is shown by the following indicator:
  - VIRTUAL BUFFER POOLS (F) = 407.71 MB

- Independent of the page size, a control block of 128 bytes is allocated for each page in the virtual buffer pool.

  The storage required is shown by the following indicator:
  - VIRTUAL BUFFER POOL CONTROL BLOCKS (G) = 12.74 MB
Hiperpools are not allocated in the DBM1 address space but storage is allocated inside the DBM1 address space for the 56-byte hiperpool control block associated with each hiperpool buffer.

The storage required is shown by the following indicator:

- HIPERPOOL CONTROL BLOCKS (H) = 0 MB (hiperpools are not used here)

If you use data spaces, additional storage is required in the DBM1 address space. A control block of 128 bytes is allocated for each data space buffer defined. The lookaside buffers also utilize a small amount of DBM1 virtual storage.

The storage required is shown by the following indicators:

- DATASPACE BP CONTROL BLOCKS (I) = 0 MB (data spaces are not used here)
- DATASPACE LOOKASIDE BUFFER (J) = 0 MB

In our example, virtual buffer pools and control structures use a total of 420.45 MB, which represents 86% of the total DBM1 storage.

Beside the virtual storage required for the virtual pools and hiperpools, other components can be large consumers of virtual storage in the DBM1 address space depending on the workload and system parameter settings:

- The EDM pool containing active and skeleton plans and packages, dynamic statements, and Database Descriptors (DBDs). The maximum size is specified by the system parameter EDMPOOL in DSNZPARM.
  
  The storage used is shown by the following indicator:
  
  - EDM POOL (K) = 19.53 MB

- The RID pool used for list prefetch, multiple index access processing, and hybrid joins. The maximum size is specified by the system parameter MAXRBLK in DSNZPARM.
  
  The storage used is shown by the following indicator:
  
  - RID POOL (L) = 0.57 MB

- The Local Dynamic Statement Cache (DSC), which size is indirectly determined by the system parameter MAXKEEPD in DSNZPARM.
  
  The storage used is shown by the following indicators:
  
  - LOCAL DYNAMIC STMT CACHE CTL BLKS (M) = 0.99 MB
  - LOCAL DYNAMIC STMT CACHE STMT POOL (N) = 0 MB (DSC is not used here)

- The compression dictionary for each page set/partition that is opened. The size of a compression dictionary can be up to 64 KB.
  
  The storage required is shown by the following indicator:
  
  - COMPRESSION DICTIONARY (O) = 0 MB (compression is not used here)

- TOTAL AGENT SYSTEM STORAGE (V) = 17.43 MB
  
  It is calculated as 50 KB times the sum of the internal system agents (prefetch engines, deferred write engine, castout engines). In this case it is 50 * (39+300+0+0+0+0)=17 MB.
Storage is also allocated on a per-thread basis for various internal DB2 areas:

- TOTAL AGENT LOCAL STORAGE (P) = 26.64 MB
- RDS OP POOL (Q) = 3.32 MB
- BUFFER & DATA MANAGER TRACE (R) = 6.59 MB
- TOTAL GETMAINED STACK STORAGE (E) = 7.78 MB

The total storage allocated for threads is summarized by the indicator TOTAL STORAGE FOR ALL THREADS (S). In our example, threads use 44.33 MB, which represents 9% of the total DBM1 storage. As the TOTAL NUMBER OF ACTIVE USER THREADS (T) is 75, each thread consumes 0.6 MB of DBM1 storage.

Table 3-2 summarizes the major consumers of virtual storage allocated in the DBM1 address space. For example 1: transaction workload running in a non-data sharing environment.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Storage usage</th>
<th>% of total DBM1 storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual buffer pools</td>
<td>420.45 MB</td>
<td>86.0%</td>
</tr>
<tr>
<td>Thread storage</td>
<td>44.33 MB</td>
<td>9.1%</td>
</tr>
<tr>
<td>EDM pool</td>
<td>19.53 MB</td>
<td>4.0%</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>0.9%</td>
</tr>
</tbody>
</table>

The DBM1 STORAGE STATISTICS section of the DB2 PM statistic long report also indicates the size of the storage cushion. In our example, STORAGE CUSHION (U) = 91.85 MB.

The storage cushion is an internal threshold. 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 the service, so you should try to avoid frequent occurrences.

The storage cushion size is determined by three system parameters in DSNZP ARM: the maximum number of open data sets (DSMAX); the maximum number of allocated threads (CTHREAD); and the maximum number of distributed threads (MAXDBAT). Be careful not to over-configure these values.

**Example 2: Transaction workload in a data sharing environment**

The same workload (IRWW) was run in a data sharing environment. Figure 3-2 is an extract of the DB2 PM statistics long report for one of the two members of the data sharing group. It corresponds to the DBM1 STORAGE STATISTICS section.
In a data sharing environment, some additional storage is required:

- A castout buffer of 128 KB is required for each castout engine. The storage required is shown by the following indicator:
  - CASTOUT BUFFERS (A): 11.25 MB
  
  This indicator is derived as: NUMBER OF CASTOUT ENGINES (B) * 128 / 1024

Table 3-3 summarizes the major consumers of virtual storage allocated in the DBM1 address space for example 2 where the transaction workload is running in a data sharing environment.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Storage usage</th>
<th>% of total DBM1 storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual buffer pools</td>
<td>420.45 MB</td>
<td>83.9%</td>
</tr>
<tr>
<td>Thread storage</td>
<td>44.56 MB</td>
<td>8.9%</td>
</tr>
<tr>
<td>EDM pool</td>
<td>19.53 MB</td>
<td>3.9%</td>
</tr>
<tr>
<td>Castout buffers</td>
<td>11.25 MB</td>
<td>2.2%</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>1.1%</td>
</tr>
</tbody>
</table>
Example 3: Single parallel query
A single parallel query was run in a non-data sharing environment. Figure 3-3 is an extract of the DB2 PM statistics long report. It corresponds to the DBM1 STORAGE STATISTICS section.

Figure 3-3 STORAGE STATISTICS report layout - Single parallel query

When parallelism is used, some additional storage is required for the PIPE MANAGER SUB POOL (A). In our example, it represents only 0.22 MB, but this pool can rapidly grow if parallelism is heavily used.

Table 3-4 summarizes the major consumers of virtual storage allocated in the DBM1 address space for example 3 where a single parallel query is running in a non data sharing environment.

Table 3-4 Example 3: Virtual storage consumers

<table>
<thead>
<tr>
<th>Resource</th>
<th>Storage usage</th>
<th>% of total DBM1 storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtual buffer pools</td>
<td>241.70 MB</td>
<td>93.4%</td>
</tr>
<tr>
<td>Thread storage</td>
<td>9.49 MB</td>
<td>3.7%</td>
</tr>
<tr>
<td>EDM pool</td>
<td>4.88 MB</td>
<td>1.9%</td>
</tr>
<tr>
<td>Others</td>
<td></td>
<td>1.0%</td>
</tr>
</tbody>
</table>
3.1.4 DB2 PM record trace: storage manager pool summary

The records from IFCID 0225 can also be accessed using DB2 PM record trace command. An example of this command is listed in Example 3-2.

Example 3-2  RECTRACE command

```
RECTRACE
  TRACE {
    LEVEL(LONG)
    INCLUDE(IFCID(225))
  }
EXEC
```

Figure 3-4 shows an example of DB2 PM long record trace for IFCID 0225. This record trace corresponds to “Example 2: Transaction workload in a data sharing environment” on page 23.

```
TOTAL GETMAINED (A) : 474331512  TOTAL RDS OP POOL : 3928064
TOTAL COMPRESS DICTIONARY : 0  TOTAL AGENT SYSTEM : 40980480
AVAILABLE : 1258553344  RESERVED FOR MUST-COMPLETE (E) : 40980480
AGENT LOCAL POOL : 25894912  TOTAL STACK (C) : 9154560
AMOUNT FOR MVS USE (F) : 11733160  LOCAL DYNAMIC STMT CACHE POOL : 4096
RID POOL : 675840  PIPE MANAGER SUBPOOL : 9154560
LOCAL DYNAMIC STMT CACHE CTL BLOCKS : 1036288  TOTAL VARIABLE (D) : 41271296
BUFFER & DATA MANAGER TRACE TBL : 6909952
NO ACTIVE ALLIED THREADS : 75
NO CASTOUT ENGINES : 90  NO DEFERRED WRITE ENGINES : 2
NO GBP WRITE ENGINES : 128  NO PREFETCH ENGINES : 42
NO P-LOCK/NOTIFY EXIT ENGINES : 27  CUSHION WARNING TO CONTRACT (G) : 40980480
```

Figure 3-4  Storage manager pool summary - IRWW in data-sharing

Most of the indicators are the same as in the DB2 PM statistics report, others can be derived:

- From the record trace, the total DBM1 storage can be derived as follows:
  
  \[
  \text{Total DBM1 storage} = \text{TOTAL GETMAINED (A)} + \text{TOTAL FIXED (B)} + \text{TOTAL STACK (C)} + \text{TOTAL VARIABLE (D)}
  \]

- From the record trace, the storage cushion size can be derived as follows:

  \[
  \text{Storage cushion} = \text{RESERVED FOR MUST-COMPLETE (E)} + \text{AMOUNT FOR MVS USE (F)} + \text{CUSHION WARNING TO CONTRACT (G)}
  \]

3.2 Other new IFCIDs

A number of IFCIDs are added in DB2 V7. Most of them are described in DB2 for z/OS and OS/390 Version 7 Performance Topics, SG24-6129. Some are introduced by recent APARs and are described in this section.

Table 3-5 sums up all the new IFCIDs.

Table 3-5  New IFCIDs

<table>
<thead>
<tr>
<th>IFCID</th>
<th>Trace Type</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0217</td>
<td>Global</td>
<td>10</td>
<td>It records detailed information about storage usage in the DBM1 address space.</td>
</tr>
</tbody>
</table>
3.2.1 IFCID 0234

This record is introduced by APAR PQ47973. It returns the authorization ID information for the calling agent. It is always active. No start command is necessary to make it active and available for the IFI READS command. It will not respond to start or stop trace commands and cannot be turned off. See A.2, “DB2 for OS/390 V6 APARs” on page 110 for the list of performance related DB2 V6 APARs.

3.2.2 IFCID 0334

This IFCID records first failure data capture records for errors that are detected by DRDS. DRDS generates this record when a protocol violation is detected when a CNTQRY command is received.

3.3 Changed IFCIDs

In this section we mention some changes to existing IFCIDs.

<table>
<thead>
<tr>
<th>IFCID</th>
<th>Trace Type</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0219</td>
<td>Audit Performance</td>
<td>8, 10</td>
<td>It records the use of LISTDEFs by utilities.</td>
</tr>
<tr>
<td>0220</td>
<td>Audit Performance</td>
<td>8, 10</td>
<td>It records information about dynamically allocated utility output data sets.</td>
</tr>
<tr>
<td>0225</td>
<td>Statistics</td>
<td>6</td>
<td>Summary of storage usage in DBM1 address space.</td>
</tr>
<tr>
<td>0234</td>
<td>always active</td>
<td></td>
<td>It returns authorization ID information for the calling agent.</td>
</tr>
<tr>
<td>0319</td>
<td>Audit</td>
<td>8</td>
<td>It records Kerberos security translation of user IDs.</td>
</tr>
<tr>
<td>0334</td>
<td>Performance</td>
<td>32</td>
<td>It records first failure data capture records for errors detected by DRDS</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IFCID</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0219</td>
<td>8, 10</td>
<td>It records the use of LISTDEFs by utilities.</td>
</tr>
<tr>
<td>0220</td>
<td>8, 10</td>
<td>It records information about dynamically allocated utility output data sets.</td>
</tr>
<tr>
<td>0225</td>
<td>6</td>
<td>Summary of storage usage in DBM1 address space.</td>
</tr>
<tr>
<td>0234</td>
<td></td>
<td>It returns authorization ID information for the calling agent.</td>
</tr>
<tr>
<td>0319</td>
<td>8</td>
<td>It records Kerberos security translation of user IDs.</td>
</tr>
<tr>
<td>0334</td>
<td>32</td>
<td>It records first failure data capture records for errors detected by DRDS</td>
</tr>
</tbody>
</table>

3.2.1 IFCID 0234

This record is introduced by APAR PQ47973. It returns the authorization ID information for the calling agent. It is always active. No start command is necessary to make it active and available for the IFI READS command. It will not respond to start or stop trace commands and cannot be turned off. See A.2, “DB2 for OS/390 V6 APARs” on page 110 for the list of performance related DB2 V6 APARs.

3.2.2 IFCID 0334

This IFCID records first failure data capture records for errors that are detected by DRDS. DRDS generates this record when a protocol violation is detected when a CNTQRY command is received.

3.3 Changed IFCIDs

In this section we mention some changes to existing IFCIDs.

IFCID 0002, 0003, 0148: page P-lock counters

New counters are used for both Statistics and Accounting traces to record detail on page P-lock requests in a data sharing environment. These counters record the following:

- Number of page P-lock requests for space map pages
- Number of page P-lock requests for data pages
- Number of page P-lock requests for index leaf pages
- Number of page P-lock unlock requests
- Number of page P-lock suspensions for space map pages
- Number of page P-lock suspensions for data pages
- Number of page P-lock suspensions for index leaf pages

The Statistics Trace also includes the following additional counters:

- Number of page P-lock negotiations for space map pages
- Number of page P-lock negotiations for data pages
- Number of page P-lock negotiations for index leaf pages
This information is reported by DB2 PM V7 when APAR PQ46636 is applied. It can be found in the Group Buffer Pool Activity block of both DB2 PM statistics and accounting reports. Figure 3-5 is an extract of a DB2 PM statistics report. It shows an example of these new counters.

<table>
<thead>
<tr>
<th>GROUP TOT4K CONTINUED</th>
<th>QUANTITY</th>
<th>/SECOND</th>
<th>/THREAD</th>
<th>/COMMIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAGE P-LOCK LOCK REQ</td>
<td>115.7K</td>
<td>192.77</td>
<td>N/C</td>
<td>1.49</td>
</tr>
<tr>
<td>SPACE MAP PAGES</td>
<td>43724.00</td>
<td>72.88</td>
<td>N/C</td>
<td>0.56</td>
</tr>
<tr>
<td>DATA PAGES</td>
<td>0.00</td>
<td>0.00</td>
<td>N/C</td>
<td>0.00</td>
</tr>
<tr>
<td>INDEX LEAF PAGES</td>
<td>71929.00</td>
<td>119.89</td>
<td>N/C</td>
<td>0.92</td>
</tr>
<tr>
<td>PAGE P-LOCK UNLOCK REQ</td>
<td>114.9K</td>
<td>191.45</td>
<td>N/C</td>
<td>1.47</td>
</tr>
<tr>
<td>PAGE P-LOCK LOCK SUSP</td>
<td>2182.00</td>
<td>3.64</td>
<td>N/C</td>
<td>0.03</td>
</tr>
<tr>
<td>SPACE MAP PAGES</td>
<td>2017.00</td>
<td>3.36</td>
<td>N/C</td>
<td>0.03</td>
</tr>
<tr>
<td>DATA PAGES</td>
<td>0.00</td>
<td>0.00</td>
<td>N/C</td>
<td>0.00</td>
</tr>
<tr>
<td>INDEX LEAF PAGES</td>
<td>165.00</td>
<td>0.28</td>
<td>N/C</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGE P-LOCK LOCK NEG</td>
<td>1938.00</td>
<td>3.23</td>
<td>N/C</td>
<td>0.02</td>
</tr>
<tr>
<td>SPACE MAP PAGES</td>
<td>1898.00</td>
<td>3.16</td>
<td>N/C</td>
<td>0.02</td>
</tr>
<tr>
<td>DATA PAGES</td>
<td>0.00</td>
<td>0.00</td>
<td>N/C</td>
<td>0.00</td>
</tr>
<tr>
<td>INDEX LEAF PAGES</td>
<td>40.00</td>
<td>0.07</td>
<td>N/C</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 3-5  Page P-lock counters

**IFCIDs 0003, 0147, 0148: global contention**

The Class 3 wait time for global contention (data sharing only) was reported as a single value in Version 6.

In Version 7 this is now broken down into pairs of elapsed time and event counters, as follows:

- Waits for parent L-locks (database, table space, table, or partition)
- Waits for child L-locks (page, or row)
- Waits for other L-locks
- Waits for page set and partition P-locks
- Waits for page P-locks
- Waits for other P-locks

This information is reported by DB2 PM V7 when APAR PQ46636 is applied. It can be found in the new Global Contention L-Locks and Global Contention P-Locks blocks of DB2 PM accounting report. Figure 3-6 shows an example of these new counters.

<table>
<thead>
<tr>
<th>GLOBAL CONTENTION</th>
<th>L-LOCKS AVERAGE TIME</th>
<th>AV. EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-LOCKS</td>
<td>0.002396</td>
<td>0.05</td>
</tr>
<tr>
<td>PARENT (DB,TS,TAB,PART)</td>
<td>0.000000</td>
<td>0.00</td>
</tr>
<tr>
<td>CHILD (PAGE,ROW)</td>
<td>0.001832</td>
<td>0.03</td>
</tr>
<tr>
<td>OTHER</td>
<td>0.000000</td>
<td>0.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>GLOBAL CONTENTION</th>
<th>P-LOCKS AVERAGE TIME</th>
<th>AV. EVENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-LOCKS</td>
<td>0.000000</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGESET/PARTITION</td>
<td>0.000000</td>
<td>0.00</td>
</tr>
<tr>
<td>PAGE</td>
<td>0.000000</td>
<td>0.00</td>
</tr>
<tr>
<td>OTHER</td>
<td>0.000564</td>
<td>0.03</td>
</tr>
</tbody>
</table>

Figure 3-6  Global contention
3.4 DB2 PM workstation interface enhancements

Two major enhancements have been introduced by service for DB2 PM.

3.4.1 New Workstation Online Monitor functions

PTF UQ58284 for APAR PQ51708 introduces new DB2 PM Workstation Online Monitor functions:

- SQL activity report
  - Allows easy access to SQL activity reports
  - Display of the report in a browser-like view
  - Full control of report generation from Workstation Client

- Locking conflicts
  Sysplex-wide display of locks on resource and thread level
  Possibility to cancel thread and access thread details

- Thread holding resources
  - Display of resources locked by a specific thread

- Multiple Sort and Qualify for thread display
  Enhanced filtering to handle large amount of data
  Multiple sort to order data highly customizable

- Improved handling of Thread, Statistics and System Parameters

- Improved usability according customer requirements, designed by accredited test lab for the testing of ergonomics of software IT products

- Reduced maintenance effort on customer side as new DB2 performance counter support can easily be distributed

A new edition of *DB2 PM Version 7 Using the Workstation Online Monitor*, SC27-0859-01 is available to all users of the workstation interface, and can be found at:


or

http://www.ibm.com/software/data/db2imstools/performmgt.html

3.4.2 The Performance Warehouse function

Another large change for DB2 PM is in APAR PQ57168. It delivers the Performance Warehouse function. Performance Warehouse lets you define, schedule, and run processes that do the following:

- Automate the creation of reports
- Automate the conversion and loading of these reports, or other save or file data sets that exist on the host, in a performance database.

Furthermore, you can analyze the data in the performance database by using rules defined by you. The PWH client is shipped with the Workstation Online Monitor DB2 PM V7. The new edition of the standard DB2 PM documentation, available from the Web sites listed above, describes the new functionality.

For more information on this function and other recent changes to DB2 PM watch out for the new redbook *DB2 Performance Expert for z/OS*, SG24-6867.
Java support

Since DB2 started to support JAVA, a large number of enhancements were made to improve its performance. In this chapter we describe the latest enhancements to JDBC and SQLJ native environments, starting with a brief description of JDBC and SQLJ, and the differences between them.

We also present some guidelines for high performance using JDBC/SQLJ, covering application, system, and environment aspects. Finally, we report some performance measurements, and we list hints and tips on analyzing performance. Most of these considerations apply to stored procedures implemented with the JDBC driver as well.

We examine the following topics:

- JDBC and SQLJ
- SQLJ versus JDBC
- Enhancements to JDBC/SQLJ driver
- Guidelines for high performance when using JDBC/SQLJ
- Analyzing performance
- Performance results
4.1 JDBC and SQLJ

In this section we provide an overview of JDBC and SQLJ, and compare the advantages and disadvantages of each.

4.1.1 JDBC overview

The JDBC is a standard API for Java applications to connect from Java to relational databases. JDBC consists of a set of classes and interfaces written in Java that provide a standard API for Java application developers to implement database applications. JDBC offers portability across platforms and database systems. JDBC supports the use of dynamic SQL, so it can deal with situations where you do not know the table and column names at the time the application is written. On the other hand, it has the performance disadvantage that the statement is prepared at run time.

Figure 4-1 shows the components of a JDBC implementation. The primary function of the JDBC driver manager component is to connect the Java application to the correct JDBC driver, and then it is no longer used anymore in this connection.

![JDBC Diagram]

Figure 4-1  JDBC

Basically, a JDBC driver is responsible for these major tasks:

- Establishes a connection with a database
- Sends SQL statements to the database
- Processes the results

Example 4-1  Simple JDBC application

```java
Connection con = DriverManager.getConnection(  
"jdbc:db2:sample", "login", "password");
PreparedStatement stmt = con.prepareStatement(  
"SELECT a, b, c FROM Table1");
ResultSet rs = stmt.executeQuery();
while (rs.next()) {
    int x = getInt("a");
    String s = getString("b");
    float f = getDate("c");
}
```

.
Types of JDBC driver
The JDBC drivers can be classified into four types, as depicted in Figure 4-2, depending on how they are implemented:

- **Type 1 – JDBC-ODBC Bridge**: This is a transition solution, which requires an ODBC driver to work.
- **Type 2 – Native API**: This is not 100% Java, but converts JDBC calls to database specific API calls. Part of the JDBC is implemented in Java that uses the Java Native Interface (JNI) to call database specific API.
- **Type 3 – Net Protocol**: 100% Java. This converts JDBC calls into DBMS-independent network protocol.
- **Type 4 – Native Protocol**: This converts JDBC calls directly into the network protocol used by the DBMS. In case of DB2 a Type 4 driver converts JDBC into DRDA. The client can directly access the DB2 system via DRDA.

Type 3 and Type 4 drivers must route through a network layer, so they are not efficient for local JDBC connectivity. In this case, it is better to use a Type 2 driver.

![Figure 4-2 Types of JDBC Drivers](image-url)
4.1.2 SQLJ overview

SQLJ provides support for embedded SQL in Java applications. SQLJ was initially developed by Oracle, Tandem, and IBM to complement the dynamic SQL JDBC model with a static SQL model. It has been accepted by ANSI and ISO. In general, Java applications use JDBC for dynamic SQL, and SQLJ for static SQL, but it is possible to mix both Types in an application. Example 4-2 provides a comparison of the two manners of Java coding.

Example 4-2   Same program using JDBC and SQLJ

**JDBC**

```java
java.sql.PreparedStatement ps =
con.prepareStatement("SELECT ADDRESS FROM EMP WHERE NAME=?");
ps.setString(1, name);
java.sql.ResultSet rs = ps.executeQuery();
rs.next();
addr = rs.getString(1);
rs.close();
```

**SQLJ**

```sql
#sql [con] { SELECT ADDRESS INTO :addr FROM EMP
WHERE NAME=:name
};
```

The use of static SQL allows authority, syntax, access strategy, and logic checking to be done at SQL BIND time. This unique ability provides performance improvements for applications that repeatedly use the same SQL.

The SQLJ specification consists of three parts:

- **Database Languages – SQLJ – Part 0**: Object Language Bindings (SQL/OLB) is also known as SQLJ Part 0. It was approved by ANSI in 1998, and it specifies the SQLJ language syntax and semantics for embedded SQL statements in a Java application.

- **Database Languages – SQLJ – Part 1**: SQL Routines using the Java Programming Language was approved by ANSI in 1999. It specifies extensions that define the installation of Java classes in an SQL database, and the invocation of static methods as stored procedures.

- **Database Languages – SQLJ – Part 2**: SQL Types using the Java Programming Language is under development. It specifies extensions for accessing Java classes as SQL user-defined types.

For the general SQLJ information, visit the SQLJ Web site:

http://www.sqlj.org

The DB2 for OS/390 implementation of SQLJ includes support for the following portions of the specification:

- Part 0
- The ability to invoke a Java static method as a stored procedure, which is in Part 1

Figure 4-3 shows an overview of the process involved in deploying an SQLJ application. The Translator receives a SQLJ (.sqlj) file and performs some semantic checking, generates calls to SQLJ run time library and produces Java (.java) and Profile (.ser) files. The Customizer is optional, vendor-specific, and it updates the profiles.
For DB2, the Customizer precompiles SQL and generates a package that needs to be bound to DB2. The Java interpreter receives the JAVA file and generates the CLASS (.class) files (bytecode) that make calls to the run time environment. Run time environment performs the SQL operations. Most of the vendors use JDBC (dynamic SQL) underneath, but in DB2 the SQL is executed statically.

### 4.2 SQLJ versus JDBC

The JDBC and SQLJ APIs are both widely accepted open industry standards. However, each has its own set of advantages and disadvantages. We examine the reasons to use either of them.

#### 4.2.1 Reasons to use SQLJ

In this section we list the advantages of using SQLJ.

**Less complex and more concise than JDBC**

SQLJ source programs are smaller than equivalent JDBC programs because certain code that the programmer must include in JDBC programs is generated automatically by SQLJ. In SQLJ programs, you can embed Java host expressions in SQL statements, while JDBC requires a separate call statement for each bind variable, and specifies the binding by position number. SQLJ has a significant advantage over JDBC in that SQLJ supports the singleton SELECT, whereas JDBC does not.
**Better performance**

In general, an SQLJ statement runs quicker than an equivalent JDBC statement, since SQLJ is not prepared at run time. JDBC presents a string containing an SQL statement to the database at run time (which may have just been constructed within the Java application). The first time DB2 gets to see the SQL is when it is actually executed, so all JDBC calls, by their very nature, consist of dynamic SQL. When DB2 receives the SQL statement, it must perform a number of steps to prepare the statement before it is able to execute it (including syntax checking, authorization checking, and access path selection), and this can often require longer to perform than the actual SQL itself.

SQLJ is able to use static SQL by actually embedding the SQL statements within the application code.

The SQLJ program preparation process extracts this SQL and binds it against the database, allowing DB2 to perform all of the checks and access the path selection as a once only process. At run time, DB2 uses the pre-prepared access plan and is able to immediately execute the SQL (see Figure 4-4).

![Dynamic versus Static SQL at execution time](image)

Figure 4-5 shows the result of a simple SQL performance comparison between JDBC and SQLJ. Using JDBC, the SQL statement is pre-loaded into the dynamic statement cache and is always found in memory there during the measurements. The measured SQL statements are:

- Open - 4 fetch - close, selecting 4 rows containing columns of different data types
- 4 inserts, inserting 4 rows containing columns of different data types
The results show improvements of 12% for the inserts, 50% for open/fetch/close, 70% for singleton selects. Just as a reminder: JDBC implements singleton select as open-1 fetch-close. The 100% dynamic statements cache (DSC) hit, assumed in these measurements for ease of comparison, is not always reached, and it represents the smallest obtainable improvement. A more realistic assumption of 95% hit, or less, could lead to even higher percentage of improvement for SQLJ versus JDBC.

Figure 4-5   SQLJ versus JDBC performance

**Users can be authorized for access to programs, not tables**
SQLJ provides the advantages of static SQL authorization checking. With SQLJ, the authorization ID under which SQL statements execute is the plan or package owner. DB2 checks table privileges at bind time. Because JDBC uses dynamic SQL, the authorization ID under which SQL statements execute is not known until run time, so no authorization checking of the table privileges can occur until run time.

**Optional SQL checking before run time**
SQLJ can do data type checking during the program preparation process to determine whether table columns are compatible with Java host expressions. JDBC passes values to and from SQL tables without compile-time data type checking. Also, in JDBC the SQL syntax is only checked at run time, while in SQLJ it is checked during the preparation process.

### 4.2.2 Reasons to use JDBC

SQLJ requires more steps in the application build process, and some development tools may not support these additional steps. Also, in some cases, more flexibility is needed to dynamically build SQL requests at run time.

### 4.2.3 SQLJ and JDBC inter-operability

It is possible to mix SQLJ and JDBC access in the same application; for example, if you need to selectively use dynamic SQL. In this case, SQLJ and JDBC can share the same JDBC connection. You can use SQLJ iterator to retrieve data from JDBC result sets or generate JDBC result sets from SQLJ iterators. See Chapter 2 of *DB2 UDB for OS/390 and z/OS Version 7 Programming Guide and Reference for Java*, SC26-9932 for details.
4.3 Enhancements to JDBC/SQLJ driver

DB2 V7 provides the following implementations of JDBC and SQLJ:

- The SQLJ/JDBC driver with JDBC 1.2 support, which is fully compliant with the JDBC 1.2 and SQLJ – Part 0 specification. To use this version of JDBC, you need the Java Development Kit (JDK) for OS/390, Version 1.1.6 or higher.

- The SQLJ/JDBC driver with JDBC 2.0 support, which is fully compliant with the JDBC 1.2 and SQLJ – Part 0 specification and includes most of the functions of the JDBC 2.0 specification. To use this version of JDBC, you need the Java 2 Software Development Kit (SDK, previously called JDK) for OS/390, Version 1.3 or higher.

You select a version of the JDBC driver by specifying the associated file name in your CLASSPATH environment variable.

Since DB2 V7 was generally available, several performance enhancements were made to the JDBC/SQLJ drivers. The JDBC 2.0 at DB2 V7 GA driver performs equivalently to JDBC 1.2, the enhancements were only implemented in JDBC 2.0 driver. They were not retrofitted to JDBC 1.2. The improvements in JDBC/SQLJ 2.0 are:

- **Context switching performance improvements:**
  - When multiple context is enabled, a Java program can maintain multiple java.sql.Connection objects simultaneously, and each Connection object is related to a unique context (DB2 thread).
  - Potentially, each SQL statement in the Java program can be executed on a different context. Now the JDBC driver checks if the next SQL statement is using a different context before calling the context switch. For more details on context, see “JDBC and SQLJ multiple OS/390 context support” in Chapter 6 of DB2 UDB for OS/390 and z/OS Version 7 Programming Guide and Reference for Java, SC26-9932

- **Lower cost for column processing:**
  - The design of how columns are moved to and from DB2 to the Java program is re-architected to dramatically improve performance (getxxx and setxxx methods).
  - The overall cost is now mainly determined by the cost of the constructor call of the returned object type. The overall impact of this performance improvement is dependent on the number of columns processed in the application and the Java data types used; it can be huge in some cases.

- **Fewer Java Native Interface (JNI) crossing:**
  - This allows Just in Time compiler (JIT) to be more effective by optimizing the use of JNI calls.

- **Presume abort logging in RRS Attach:**
  - This makes logging more efficient.

- **Java Virtual Machine (JVM) improvements:**
  - Code page conversion are now implemented as intrinsics. This improves conversion performance dramatically.
  - SDK 1.3 JNI callback optimization

- **SQLJ optimization for UPDATE or DELETE WHERE CURRENT OF (positioned update/delete):**
  - Before this optimization, positioned update/delete were always executed as dynamic SQL because DB2 requires that the cursor and the related update/delete are in the same package.
In SQLJ this is not required; an iterator can be passed as an argument to another
method to perform positioned update/delete. Therefore, the cursor can be in one
package and the update/delete in a different.

Now, SQLJ runtime checks, before executing the update/delete statements, if the
related cursor is in the same package. If this is the case, the positioned update/delete
will be executed as static SQL. Otherwise, the positioned update/delete will be
executed as dynamic SQL as before. This function is introduced with APAR PQ51847.

Table 4-1 shows a list of APARs with the JDBC/SQLJ enhancements.

<table>
<thead>
<tr>
<th>APAR</th>
<th>Enhancements</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ51847</td>
<td>Support for program versioning. Provides the same functionality as the DB2 precompiler “VERSION” option.</td>
</tr>
<tr>
<td></td>
<td>Support for true STATIC positioned update/delete processing for customized SQLJ applications when the positioned update/delete is performed within the same program as the SELECT.</td>
</tr>
<tr>
<td></td>
<td>Support for LOBs.</td>
</tr>
<tr>
<td></td>
<td>Performance enhancement for character conversion due to better caching of ByteToChar converter objects.</td>
</tr>
<tr>
<td></td>
<td>Support of JDBC 2.0 Batch Update. The following APIs are now supported:</td>
</tr>
<tr>
<td></td>
<td>Statement.addBatch()</td>
</tr>
<tr>
<td></td>
<td>Statement.clearBatch()</td>
</tr>
<tr>
<td></td>
<td>Statement.executeUpdate()</td>
</tr>
<tr>
<td></td>
<td>PrepareStatement.addBatch()</td>
</tr>
<tr>
<td></td>
<td>Implementations and enhancements for many of the JDBC 2.0 APIs:</td>
</tr>
<tr>
<td></td>
<td>Connection.createStatement (int type, int concurrency)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.othersDeletesAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.othersInsertsAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.othersUpdatesAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.ownDeletesAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.ownInsertsAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.ownUpdatesAreVisible (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.deletesAreDetected (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.insertsAreDetected (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.updatesAreDetected (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.supportsResultSetType (int type)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.supportsResultSetConcurrency (int type, int concurrency)</td>
</tr>
<tr>
<td></td>
<td>DatabaseMetadata.supportsBatchUpdate ()</td>
</tr>
<tr>
<td></td>
<td>PreparedStatement.setCharacterStream ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.getCharacterStream ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.getConcurrency ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.getFetchDirection ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.getFetchSize ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.getType ()</td>
</tr>
<tr>
<td></td>
<td>ResultSet.setFetchSize (int rows)</td>
</tr>
<tr>
<td></td>
<td>ResultSetMetaData.getColumnClassName (int column)</td>
</tr>
<tr>
<td></td>
<td>Statement.getFetchDirection ()</td>
</tr>
<tr>
<td></td>
<td>Statement.getResultSetConcurrency ()</td>
</tr>
<tr>
<td></td>
<td>Statement.getResultSetType ()</td>
</tr>
</tbody>
</table>
4.4 Guidelines for high performance when using JDBC/SQLJ

These guidelines can be divided into three groups:

- Designing the application
- Identifying the required system levels
- Environment tuning

4.4.1 Design guidelines for applications

In this section we list the recommendations about coding and preparation of SQLJ and JDBC program.

Map Java data types to DB2 data types

For optimal performance, we recommend that you map the Java data types used to the SQL column data types. The primary reason for this is to provide for efficient predicate processing: indexable and Stage 1. The other reason is to minimize data conversion cost.

Table 4-2 provides the recommended mapping between Java data types and SQL column data types.
Table 4-2  Mapping DB2 data types to Java data types

<table>
<thead>
<tr>
<th>DB2 data type</th>
<th>Java data type</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMALLINT</td>
<td>short, boolean</td>
<td>No direct mapping for bit in DB2</td>
</tr>
<tr>
<td>INTEGER</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>float</td>
<td>Single precision</td>
</tr>
<tr>
<td>DOUBLE, FLOAT</td>
<td>double</td>
<td>Double precision</td>
</tr>
<tr>
<td>DECIMAL(p, s) or NUMERIC(p, s)</td>
<td>java.math.BigDecimal</td>
<td>with p=precision, s=scale keep scale and precision in Java</td>
</tr>
<tr>
<td>CHAR, VARCHAR, GRAPHIC, VARGRAPHIC</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>CHAR, VARCHAR FOR BIT DATA</td>
<td>Byte[]</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>java.sql.Date</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>java.sql.Time</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>java.sql.Timestamp</td>
<td></td>
</tr>
</tbody>
</table>

The JDBC/SQLJ driver uses getxxx() methods to retrieve the value of a column from the database. JDBC API defines that each getxxx() method returns a matching Java object. For example getString() returns a String object. The processing cost of each getxxx() method is mainly determined by the cost of the object constructor call. Returning values of Java native data type like Integer is much cheaper than returning complex objects like a Timestamp object.

Figure 4-6 shows the relative cost of all getxxx() methods compared to getShort(). Retrieving a Date column is about 21 times more expensive than retrieving a short column. Based on this information the database can be designed for high performance.
JDBC API allows to use different getxxx() methods to retrieve a database column. Using a non-matching getxxx() method is syntactically correct but it causes a performance overhead per column. The overhead depends on the DB2 data type.

Figure 4-7 shows the overhead retrieving a column of a certain data type using getString() method instead of the matching getxxx().

![CPU overhead for non-matching data type](chart.png)

**Only select and update columns as necessary**

For optimal performance, it has always been recommended that DB2 applications should only select and update the columns actually required by your application. The weight of this recommendation is even stronger for Java applications. This is because of the emphasis on individual column processing within the Java programming model. The JDBC API defines that every retrieved column is returned as a Java object. Therefore, a Java object is created for every single column processed if the data type is not a primitive type in Java.

**Store integers as SMALLINT or INT**

Most of the character data is stored in the database using ASCII or EBCDIC encoding schema. Because Java runs in Unicode, the character data needs to be converted to Java Unicode representation. Numbers are not dependent on the encoding schema, so they do not store integers as character string data (CHAR and VARCHAR). Otherwise you will have the conversion overhead between EBCDIC/ASCII and UNICODE.

**Turn auto commit off**

Turn auto commit off and execute commit as required. The default setting for auto commit is on. Executing a commit after each SQL statement adds noticeable execution cost, even in a read-only environment, since it releases and deallocates locks, unregisters and registers with RRS and maintains JDBC objects.

To turn auto commit off in a Java program, specify:

```java
conn.setAutoCommit(false)
```
Use JDBC DataSource connection pooling

The new JDBC DataSource connection pooling support reduces the cost of connection to DB2 dramatically. The cost reduction was performed in two areas:

- Reusing the DB2 connection thread. A signon is driven to identify the new user. This also forces writing accounting records
- Keeping the JDBC object

Example 4-3 is a code example that shows how to define a DataSource. Usually this is done only once by the database administrator.

Example 4-3  DataSource definition

```java
//executed only once by DBA
ds = new com.ibm.db2.jcc.DB2DataSource();
ds.setDatabaseName("TESTDB");
```

Example 4-4 shows how to use a pooled connection within an application.

Example 4-4  Connection pooling

```java
//get connection from pool
Connection Conn1 = ds.getConnection("user","password");

// Turn off auto commit default
Conn1.setAutoCommit(false);

....

Conn1.close();
```

DB2 data type CHAR versus VARCHAR in database design

In SQL, string columns are padded with blanks so that the source and target have the same length at the time the predicates are evaluated. Java does not pad, so trailing blanks are considered significant characters.

If you use CHAR, and you have defined a string of CHAR(4), `ABC` is not equal to `ABC`. You have to use a Java trim() method to eliminate the trailing blanks. This is a burden on the application programmer and is an obvious source of application coding errors. In addition, this operation increases the Java CPU cost (by more than the cost of using VARCHAR). However, this cost is only incurred by Java applications (no impact on accessing the same data from CICS, QMF, etc.).

If you use VARCHAR, the in DB2 cost will be somewhat higher. This cost will be paid by all applications that access the data (Java, CICS, Cobol, QMF/TSO, etc.). The advantage is that this approach makes life easier for the Java application programmer.

The decision depends on your priorities.
Release resources
A JDBC driver maintains its own links to resources. They are released only when the resources are closed or the connection is closed. Therefore, garbage collection cannot reclaim those objects and eventually the application may be running out of JDBC resources or, even worse, out of memory. So it is important to close the following:

- **Result sets**: Otherwise, it is possible to run out of available cursors.
- **Prepared statements**: Otherwise, it is possible to run out of available cursors because closing the result sets is not sufficient. Also close them before reusing the statement handle to prepare a different SQL statement within the same connection. Use prepState.close() method.
- **Callable statements**: Otherwise, it is possible to run out of available call sections.

Run db2genJDBC
You can modify the number of DB2 cursors available for JDBC and to control cursor names by customizing the cursor properties file. After you customize it, you must run db2genJDBC.

Perform online checking with db2profc
For string data types in Java there is no concept of length. The associated SQL column data types are CHAR and VARCHAR. In order to have the predicates used for index matching, the definition in the DBRM for SQLJ (static SQL) must match the definition in the DB2 catalog in terms of data type and length. To achieve this, you must customize the SQLJ serialized profile using db2profc and specify the online checker option -online = <DB2 location name>:

```
db2profc ...-online=<db2_location_name>...
```

The online checker accesses the DB2 catalog to check JDBC/SQLJ-supported compatibility/convertibility processing and to determine the length of string columns and to add the information to the DBRM.

Be sure to run the db2profc in the same platform where the code is executed in order to have the checking performed consistently with the DB2 V7 provided code.

**Note:** If the SQLJ serialized profile is not customized the Java application will execute dynamically using JDBC. There will be no messages.

Use explicit connection context objects
If you omit the connection context object, a default connection context object is used. The default connection context object for a program is stored in a static variable of the default connection context class.

In a multi context environment (like WebSphere Application Service, WAS) the use of a default connection context is not thread-safe and must not be used. Additionally a bottleneck on usage of the default connection context can be created.

Closing the context releases the resources maintained by the connection context (like statement handles) and closes the underlying database connection. To avoid this, pass the constant KEEP_CONNECTION as an argument to indicate that the underlying database connection should be retained.

Figure 4-8 compares the constraints that may happen when using the default context.
Chapter 4. Java support

Figure 4-8  Different throughput using explicit and default context

Example 4-5 shows an example of explicit connection context declaration. See chapter 2 of DB2 UDB for OS/390 and z/OS Version 7 Programming Guide and Reference for Java, SC26-9932 for more details.

Example 4-5  Explicit connection context

```
// Connection context declaration
#sql context ctx;
...
//get context
myconn=new ctx(Conn1);
...
//use context in SQL
#sql [myconn] {set transaction isolation level read committed};

#sql [myconn] cursor001 = {SELECT FKEY,FSMALLINT,FINT
FROM WRKTB01 WHERE FKEY >= :wfkey};
...
//close context but keep database connection
myconn.close(ConnectionContext.KEEP_CONNECTION);
```

Use positioned iterators

Iterators are the SQLJ equivalent to JDBC result sets and cursors in traditional DB2 programming languages like COBOL. There are two ways to define iterators:

- By column name: named iterator
- By position in the select statement: positioned iterator

Named iterators are more convenient to use, but for optimal performance in critical applications, we recommend that you use positioned iterators. Named iterators use positioned iterators under the cover plus name hashing.

Example 4-6 shows a source code of a SQLJ program using a named iterator to retrieve the column values from a SELECT statement.
Example 4-6  Named iterator

// Named Iterator
#sql iterator testCase001a (short fkeycr, Time ftime, BigDecimal fnum);
....
short wfkeycr;
Time wftime;
BigDecimal wfnum;
...
#sql [myconn] cursor002 = {SELECT FKEY, FTIME, FNUM
FROM WRKTB01};
while (cursor002.next()) {
  wfkeycr = cursor002.Fkeycr();
  wftime = cursor002.Ftime();
  wfnum = cursor002.Fnum();
}

Example 4-7 shows how to use a positioned iterator to fetch the results of the same query.

Example 4-7  Positioned Iterator

// Positioned Iterator
#sql iterator testCase001(short, Time, BigDecimal);
....
short wfkeycr;
Time wftime;
BigDecimal wfnum;
...
#sql [myconn] cursor001 = {SELECT FKEY, FTIME, FNUM
FROM WRKTB01};
...
#sql {FETCH :cursor001 INTO :wfkeycr, :wftime, :wfnum};

4.4.2 Identifying the system levels

In this section we list recommendations regarding the system levels.

Use hardware support for IEEE floating point
This feature is available in G5 and later S/390 or zSeries processors. You must use OS/390 V2R6 and above to exploit it.

Use the correct level of the SDK
Make sure that your SDK level is at least 1.3. With SDK 1.3 you can exploit the hardware support for floating point calculations provided by the G5, G6, and zSeries processing complexes. However, a new level of SDK is available: SDK 1.3.1. This level includes:

▶ The upgrade to the SDK 1.3.1 level of the of the code incorporates the SUN 1.3.1 level of code

▶ The Persistent Reusable Java Virtual Machine technology, initially introduced in March of 2001. This technology has the following improvements to address the specific high performance requirements of short lived, fast transactional applications:
  – Substantial reduction of the overhead in initializing and terminating these types of applications implemented in the Java language
  – Making garbage collection more efficient
  – Isolation of transactions from one another
Security enhancements, some of which were previously available as the Security Toolkit technology preview includes:

- Java Cryptography Extension (IBMJCE)
- Java Cryptography Extension using CCA hardware cryptography (IBMJCE4758)
- Java Secure Sockets Extension (IBMJSSE)
- Public Key Cryptography Standards (PKCS, S/MIME)
- Java Certification Path (CertPath)
- Java Authentication and Authorization Service (JAAS)
- SAF Interfaces

Various performance enhancements in both the JVM and JIT and memory management. For certain applications, particularly those requiring quick startup, the -Xquickstart option is available.

Improvements to RAS (Reliability-Availability-Serviceability)

Cumulative service roll up through APAR PQ52781 (SDK 1.3.0 SERVICE REFRESH 10 - SR10)

You can upgrade to SDK 1.3.1 by applying APAR PQ52841.

Keep current with JDK, now SDK, releases and maintenance, since performance improvements are made in each release or PTF. For further information on Java on OS/390, see:


Keep current with JDBC driver

Major performance enhancements are based on DB2 V7, JDBC 2.0 driver. Stay current with JDBC/SQIJ maintenance. If you are installing the latest maintenance to SQIJ/JDBC (as of 01/12/2001, PQ51847 for DB2 V7) and you are using the OS/390 DB2 SQIJ/JDBC 2.0 based driver, you must have installed a minimum SDK service level:

- SDK 1.3.0 with APAR PQ50780 (SDK 1.3.0 SR9)
- or
- SDK 1.3.1 with APAR PQ54336 or above, for users of the JVM with Persistent Reusable Java Virtual Machine

4.4.3 Environment tuning

In this section we list recommendations regarding the environment tuning.

Memory usage

In a Java-JDBC/SQIJ workload, a large number of Java objects are created and released. Therefore, the tuning of the JVM heap plays an important role for the overall Java application performance.

The default initial heap size ms is 1 MB, and the default maximum heap size mx is 8 MB. This size is not sufficient in most cases.

Heap size studies in the lab and at customers show that setting the initial heap size and maximum heap size to an equal value increases the throughput dramatically. Scanning for garbage collection is not triggered so frequently and scanning of long living object again and again is reduced.
In order to have better performance, you must tune your JVM heap. Set ms and mx to an equal value. Values between 300 MB and 400 MB are common in a production environment. Figure 4-9 shows a heap size study measuring the relative throughput with three configurations of ms and mx.

![Figure 4-9  Heap size study](image)

You also must define the environment variable _cee_runopts to include the following specification:

```bash
_cee_runopts="heappools(on)"
```

This defines the way dynamic memory is used by SQLJ and JDBC applications. If you do not specify `heappools(on)`, then you incur in the considerable overhead of memory that is being repeatedly freed and allocated as objects are instantiated.

**BIND options**

Specify DYNAMICRULES(BIND) for dynamic SQL to ensure that the table access privileges of the binder are implicitly used during execution of the package. Otherwise, you will have to grant authorization over the underlying DB2 objects to the authorization ID executing the Java program. Dynamic SQL includes JDBC and cursor controlled updates and deletes in SQLJ that are not in the same package as the cursor. Use the QUALIFIER keyword of the BIND command to provide qualification for unqualified table or view names referenced in JDBC and SQLJ.
Activate dynamic SQL statement caching

For relatively simple SQL requests, the processing cost of preparing dynamic SQL statements can be very significant. To dramatically reduce the processing overhead, DB2 provides for dynamic SQL statement caching. Dynamic statement caching avoids the full cost of preparing an SQL request. Dynamic statement caching is enabled by specifying CACHEDYN=YES on the DSN6SPRM macro in DSNZPARM. When the prepared statement is found in the prepared statement cache, it is possible to make significant savings. A typical hit ratio is between 70 and 90%.

For optimal performance, we strongly recommend that you use SQLJ for database access. When using JDBC or cursor controlled update/delete that it is not issued in the same package as the cursor (see APAR PQ51847), we recommend that you turn on dynamic statement caching.

Dynamic caching also requires the same host variables be passed (parameters markers should be used), and the same authorization level.

4.5 Analyzing performance

This sections presents some tools and hints on analyzing performance of Java application.

A Java application can be executed locally or remotely, and therefore, it connects to DB2 in two ways:

- Via RRSAF or CAF for local execution (native). For example, WebSphere on OS/390 and Java application in Unix System Services
- Via DDF, for remote or distributed access. For example, WebSphere on AIX (through DB2 Connect) or a Java application using a JDBC Type 4 driver.

There are some tools available for analyzing performance, and each one of them is more appropriate than others according to the connection type:

- DB2 statistics and accounting
- DDCS trace and networks traces
- Tools exploring Java profile interface
- JDBC/SQLJ trace

DB2 statistics and accounting

DB2 statistics and accounting are used the same way as to analyze traditional applications with one remark. The JDBC/SQLJ driver executes in the OMVS (RRSAF or CAF) or in the requester (DDF), so its processing and related events are accounted in DB2 accounting CLASS 1. Remember, when analyzing an accounting report like the one in Figure 4-10, that the CLASS 1 times also include the processing cost of the JDBC/SQLJ driver.
As JDBC and SQLJ controlled update/delete that are not in the same package as the select statement are executed dynamically, you should also check the ratio hit in the dynamic statement cache by looking at the values of NOT FOUND IN CACHE/FOUND IN CACHE in the section of DYNAMIC SQL STMT of the DB2 PM accounting long report. See Figure 4-11.

```
TIMES/EVENTS          APPL(CL.1)       DB2  (CL.2)  
---------------------  ----------  ----------  
ELAPSED TIME         29:46.0232   21.512779  
NONNESTED            29:46.0232   21.512779  
STORED PROC          0.000000    0.000000  
UDF                  0.000000    0.000000  
TRIGGER              0.000000    0.000000  
CPU TIME             9:32.09001   18.512672  
AGENT                9:32.09001   18.512672  
NONNESTED            9:32.09001   18.512672  
STORED PROC          0.000000    0.000000  
UDF                  0.000000    0.000000  
TRIGGER              0.000000    0.000000  
PAR.TASKS            0.000000    0.000000  
SUSPEND TIME         N/A         0.000281  
AGENT                N/A         0.000281  
PAR.TASKS            N/A         0.000000  
                     
Figure 4-10   DB2 PM accounting report: class1 includes JDBC/SQLJ processing
```

```
DYNAMIC SQL STMT         TOTAL  
-------------------  --------  
REOPTIMIZATION      0  
NOT FOUND IN CACHE  2494  
FOUND IN CACHE      54  
IMPLICIT PREPARES   0  
PREPARES AVOIDED    0  
STMT INVALID (MAX)  0  
STMT INVALID (DDL)  0  
                      
Figure 4-11   DB2 PM accounting long report: dynamic statement cache
```

**DDCS and TCP/IP traces**

The DDCS and TCP/IP traces are used to determine problems in client Java applications connecting through DB2 Connect or Type 4 JDBC drivers. DDCS trace shows the DRDA flow, while the TCP/IP packet trace show TCP/IP activity.

**Java profiler interface**

The Java profiler interface provides a mechanism for tools like hprof and Jinsight to monitor and trace Java application execution. It requires that Just in Time (JIT) compilation to be deactivated due to its influence on the final results. So, it is not very useful to a production-like environment. It is good for a single Java application analysis. You can, for example, list the method calling path and the frequency of methods called.

Figure 4-12 shows an example of how to invoke hprof, which is provided with SDK, when running a Java program. This figure also shows a sample output.
Figure 4-12 Sample invocation and output of hprof

Jinsight is a free tool for visualizing and analyzing the execution of Java programs that are also based on the Java profiler interface. It is useful for performance analysis, memory leak diagnosis, debugging, or any task in which you need to better understand what your Java program is really doing. You can download it and get further information from the link:


JDCB/SQLJ trace

JDBC/SQLJ trace is helpful for analyzing message flow and message enter/exit. It is activated by environment var settings in the db2sqljdbc.properties file:

DB2SQLJ_TRACE_FILENAME=MYTRC.OUT
DB2SQLJ_TRACE_BUFFERSIZE=4096

The output is generated in a binary format, before it can be considered usable, it must be formatted using db2sqljtrace command to generate a text format file with extension JTRACE. Figure 4-13 provides examples of activating the trace and the formatted output.
4.6 Performance results

The measurement results presented in this section and in the previous ones were run in a
controlled laboratory environment. They are subject to the usual disclaimer and are to be
extrapolated with caution.

4.6.1 WebSphere Application Server

The tests were based in multi-thread native environment, depicted in Table 4-14, with the
following configuration:

- OS/390 V2R10
- WebSphere Application Server V3R5
  - JVM heap size = 256 MB
  - WAS connection pooling
- JDK V1.3.0 PTF UQ56153
- DB2 V7
- JDBC/SQLJ

The workload includes a JDBC and SQLJ implementation of the IBM Relational Warehouse
Workload (IRWW) and has a TPC-C-like profile. Each implementation of the workload was
executed separately and analyzed. 125 virtual clients (browsers) were driving the workload.
Figure 4-14 Tests environment

Figure 4-15 shows performance results using 2 different SQLJ/JDBC driver level and 2 different CCSIDs in DB2.

The SQLJ/JDBC 2.0 GA represent the driver level as shipped with DB2 V7, when DB2 V7 became GA. The SQLJ/JDBC 2.0, introduced with PQ54756, represent the driver level which as all maintenance applied including PQ54756.
Because the workload uses a lot of character data, the influence of code page conversion is impacting the performance of the workload. The conversion of CCSID 500 and 37 was already very optimized. The conversion of all other CCSIDs was much more expensive. Therefore, we chose to use CCSID 37 and 273 to evaluated the performance change.

Let us have a closer look at the results shown in Figure 4-15. The first four columns show that using this workload with CCSID 37: for both JDBC and SQLJ there is a slight performance improvement. We also see that the SQLJ implementation outperforms the JDBC implementation. The difference increased from 18% using the SQLJ/JDBC 2.0 GA driver to 20% using the SQLJ/JDBC 2.0 (PQ54756) driver.

The second four columns show the results when using CCSID 273. Other SBCSs would show a very similar behavior. For DBCS CCSIDs the SQLJ/JDBC 2.0 GA driver would perform slower than shown for CCSID 273, but the SQLJ/JDBC 2.0 GA driver would show nearly equivalent results for DBCS CCSIDs.

Because of the more expensive CCSID conversion, the overall throughput using CCSID 273 was 35% slower with the SQLJ/JDBC 2.0 GA driver for the SQLJ as well as the JDBC implementation.

With the SQLJ/JDBC 2.0 GA driver, all SBCSs are converted in a very optimized manner. Therefore, the IRWW SQLJ implementation of CCSID 37 and 273 performs now equivalently. This showed a 52% improvement for the SQLJ implementation using CCSID 273.

The IRWW JDBC implementation still does not perform equivalently across CCSIDs because of differences in the prepare execution. Nevertheless, the throughput increased overall by 35%.

As a summary, the highest throughput could be measured with SQLJ. Workloads processing a lot of char columns and do not use CCSID 500 or 37 will see a big overall performance improvement. The following tables give you more detail information about how column processing improved.

Table 4-3 shows the amount of performance gain in the getxxx() methods introduced by the enhancements made in the JDBC/SQLJ 2.0 driver since it went GA. To take advantage of this performance enhancement you must have PQ48383 applied. This performance gain is basically due to use of the byte array approach and improvements in JVM (code page conversion, JDBC column movement, and reduced JNI calls).
As the JDBC API getxxx() methods are used to retrieve the value of a database column, the setxxx() methods are used to store database column values. Table 4-4 shows the performance gain obtained with it.

**Table 4-3 Improvement in Java column processing cost - getxxx() methods**

<table>
<thead>
<tr>
<th>Get Function</th>
<th>Delta from JDBC 2.0 GA and JDBC with PQ48383</th>
</tr>
</thead>
<tbody>
<tr>
<td>getString (char) **</td>
<td>-91%</td>
</tr>
<tr>
<td>getString (varchar) **</td>
<td>-91%</td>
</tr>
<tr>
<td>getBigDecimal</td>
<td>-66%</td>
</tr>
<tr>
<td>getInt</td>
<td>-8%</td>
</tr>
<tr>
<td>getShort</td>
<td>-27%</td>
</tr>
<tr>
<td>getDate</td>
<td>-50%</td>
</tr>
<tr>
<td>getTime</td>
<td>-51%</td>
</tr>
<tr>
<td>getTimestamp</td>
<td>-64%</td>
</tr>
<tr>
<td>getDouble</td>
<td>-6%</td>
</tr>
<tr>
<td>getFloat</td>
<td>-6%</td>
</tr>
<tr>
<td>getBoolean</td>
<td>-31%</td>
</tr>
</tbody>
</table>

** Measured code pages other than 500 and 37

**Table 4-4 Improvements in Java column processing cost - setxxx() methods**

<table>
<thead>
<tr>
<th>Get Function</th>
<th>Delta from JDBC 2.0 GA and JDBC with PQ54756</th>
</tr>
</thead>
<tbody>
<tr>
<td>setString (char) **</td>
<td>-89%</td>
</tr>
<tr>
<td>setString (varchar) **</td>
<td>-88%</td>
</tr>
<tr>
<td>setBigDecimal</td>
<td>-10%</td>
</tr>
<tr>
<td>setInt</td>
<td>-18%</td>
</tr>
<tr>
<td>setShort</td>
<td>-48%</td>
</tr>
<tr>
<td>setDate</td>
<td>-7%</td>
</tr>
<tr>
<td>setTime</td>
<td>-7%</td>
</tr>
<tr>
<td>setTimestamp</td>
<td>-6%</td>
</tr>
<tr>
<td>setDouble</td>
<td>-27%</td>
</tr>
<tr>
<td>setFloat</td>
<td>-22%</td>
</tr>
<tr>
<td>setBoolean</td>
<td>-22%</td>
</tr>
</tbody>
</table>

** Measured code pages other than 500 and 37
Figure 4-16 shows the principal results of four levels of different performance improvements using JDBC/SQLJ.

- The first measurement shows a baseline, without the use of JDBC 2.0 connection pooling.
- The second measurement uses a different WebSphere (PTF 11) and JDBC Driver, with APAR PQ48383 applied, and, most importantly, it also uses JDBC 2.0 connection pooling, and therefore a pooled connection instead of creating a new one. The getconnection cost was significantly reduced.

- For the third measurement, dynamic statement caching was enabled and the SQL statement was found in cache. Observe that the PREPARE cost was drastically reduced, but it still is responsible for good part of the processing cost. This can be seen in Figure 4-17, where the distribution of cost for each measurement is detailed.
- The last measurement uses SQLJ instead of JDBC to access DB2. The weight of the prepare cost was 1%.
4.6.2 Summary

From the measurements, it is clear that SQLJ should be the primary choice for performance critical application. It is also clear that the Java interface with DB2 is a very dynamic area, where it is important to stay current with maintenance and evaluate the performance recommendations as they are made available.
CICS interface

In this chapter, we describe the CICS/DB2 enhancements introduced by CICS TS V2.2. We provide hints and tips about monitoring and tuning the CICS/DB2 attachment facility.

The major enhancement in the CICS TS V2.2 is the use of the Open Transaction Environment (OTE) by the CICS/DB2 attachment. This makes some performance improvements possible. We present measurements of these performance gains.

This chapter covers the following topics:

- CICS TS V2.2 — general introduction
- Tuning the CICS/DB2 attach
- CICS/DB2 attach enhancements
5.1 CICS TS V2.2

CICS TS V2.2 is now available and is recommended for all CICS customers for deployment in large-scale production. It provides important enhancements in a wide range of areas, with the major enhancements in its support for e-business applications.

CICS TS V2.2 provides a robust, high performance environment for enterprise applications written in Java. It exploits the persistent, reusable JVM, and provides a separate JVM for each application that runs in a CICS region, thereby ensuring Java applications a high degree of isolation from each other, while achieving an execution speed comparable with that of procedural languages. Java services enable applications to exploit the strengths of CICS via open Java Enterprise APIs. VisualAge for Java and WebSphere Studio may be used to develop these applications and deploy them under CICS. By supporting EJB session beans, CICS provides another dimension for application architects. See the planned redbook *Enterprise JavaBeans for z/OS and OS/390 CICS Transaction Server V2.2*, SG24-6284-01 for details.

By exploiting the WebSphere EJB Container, CICS enables construction of reusable business logic components, which are portable between CICS and WebSphere and may be deployed in either environment.

Where an EJB component needs to incorporate procedural logic modules to accomplish its business function, CICS enables this mixed-language component to run in a single execution environment with good isolation from other components, therefore improving robustness and manageability.

CICS provides an efficient and effective environment for applications written in COBOL, PL/I, C, C++ and other languages. This version strengthens application development capabilities, enables enhanced re-use of 3270 applications, and enables applications to manipulate XML directly. Functions to aid efficient application development include an enhanced 3270 bridge, an integrated CICS translator for use with COBOL and PL/I, CICS COBOL and PL/I XML application capability.

Improvements to connectivity include support for external call interface (ECI) over TCP/IP, improved CICS exploitation of TCP/IP services, connection optimization, and support for VTAM LU alias facility.

Enhancements in the area of availability include sign-on retention for persistent sessions, automatic restart of CICS data-sharing servers, and system-managed rebuild of coupling facility structures.

Standard CICS systems management facilities for the new functions are provided along with significant additions to CICS monitoring and statistics. Enhancements to CICSPlex SM include workload management for most new functions, important communications improvements, a remote MAS agent for Windows, and usability enhancements to the Web user interface.

DB2 users benefit from the CICS/DB2 interface performance improvements: DB2 group attach and the RMI purge option. The DB2 group attach facility allows a CICS customer to exploit the DB2 facility whereby the name may be specified as a DB2 Data Sharing Group instead of an explicit DB2 subsystem name.
Each group can contain one or more DB2 subsystems. This simplifies application-owning region (AOR) cloning within a sysplex and provides a greater level of availability. The CICS/DB2 Attachment Facility is enhanced to exploit CICS open transaction environment (OTE) function. When CICS TS V2.2 is connected to DB2 V6, or later, the attachment facility will execute using CICS open TCBs and will utilize DB2 function to move DB2 connections and threads between TCBs. CICS/DB2 applications coded to threadsafe standards, and defined to CICS as threadsafe, may benefit from reduced TCB switching. For threadsafe applications making heavy use of EXEC SQL, the reduced TCB switching provides a significant performance improvement.

A new Resource Manager Interface (RMI) purge option is introduced in CICS TS V2.2. This allows the writer of a task-related user exit (TRUE) to specify whether, before calling it, the RMI should defer purge and deactivate runaway.

### 5.2 Tuning the CICS/DB2 attachment

This section describes some hints and tips on performance and tuning of the CICS/DB2 attachment facility. It covers the CICS TS V1.2 and higher. Most of the concepts presented here can also be applied to CICS/ESA V4.1, CICS TS V1.3, and CICS TS V2.1. The exceptions for CICS TS V2.2 are listed in 5.3, "CICS/DB2 attachment in CICS TS V2.2" on page 79. For detailed information about the CICS/DB2 attachment see the *CICS TS for z/OS V2.1 CICS DB2 Guide*, SC34-5707.

#### 5.2.1 Architecture

The CICS/DB2 attachment facility provides CICS applications with access to DB2 data creating an overall connection between CICS and DB2. The CICS attachment management overhead runs mostly in the CICS address space and the processing to create and destroy resources it uses is done under the CICS main task control block (TCB) in the Master Subtask TCB (MSUB).

Within the overall connection between CICS and DB2, each CICS transaction that accesses DB2 needs a thread, an individual connection into DB2. Each thread runs under a thread TCB that is attached under the CICS main TCB. These thread subtask TCBs are 'daughters' of the MSUB, which itself is a 'daughter' of the CICS main TCB. Although most of the SQL processing is done under the thread TCB, some is also done under the DB2 address space. Figure 5-1 gives an overview of this architecture.

There are three types of CICS/DB2 threads:

- **Command threads**: Reserved for DB2 command processing through the DSNC transaction.
- **Entry threads**: Threads intended for high volume and highly intensive transactions, or those that require special attributes.
- **Pool threads**: Used for all transactions and commands not using an entry thread or a DB2 command thread.
CICS/DB2 definitions

Versions and releases of CICS from CICS TS V1.3 onwards no longer support running the CICS/DB2 attachment facility using the macro DSNCRCT. Instead, you must use an RCT that is defined using CICS resource definition online (RDO).

Using RDO the following resources can be defined and installed:

- **DB2CONN**: Defines the global attributes of the CICS/DB2 interface and defines the attributes of pool threads and command threads.
- **DB2ENTRY**: Defines entry threads to be used by a specific transaction or group of transactions.
- **DB2TRAN**: Defines additional transactions to be associated with a particular DB2ENTRY.

**Resource: threads and TCBs**

The main task when tuning an environment is to manage resources, in the CICS/DB2 attachment the resources are threads and TCBs.

The maximum number of thread TCBs that will be allowed in a CICS region is defined by the TCBLIMIT (equivalent of THRDMAX in RCT) parameter on the DB2CONN definition (Figure 5-2). Surprisingly, sometimes fewer TCBs will provide better throughput than more. This is because the overhead of managing large numbers of TCBs and their threads is greater than when there are fewer TCBs, and the threads associated with them have greater potential to be reused. Also, too many TCBs can cause Short on Storage (SOS) conditions in the CICS region.
In CICS TS V1.2 and above the CICS/DB2 attachment facility attaches TCBs as the work comes in and a thread is required. It does not detach TCBs when TCBLIMIT on the DB2CONN definition is reached. When its value is reached, the attachment will move TCBs between DB2ENTRYs as required. There is no need to detach a TCB in order to make it available for use on another DB2ENTRY or for use by the pool. More efficient use of the TCBs is the result and hence unused TCBs are less of a performance problem than in previous releases of the attachment.

The attachment will detach TCBs only when the attachment is shut down or the TCBLIMIT is reduced. The TCBS parameter of inquire DB2CONN reports the current number of TCBs attached. A SET DB2CONN TCBLIMIT() command can be used to raise or lower the TCBLIMIT. If TCBLIMIT is lowered, the number of TCBs does not reduce immediately but gradually. As threads are released, TCBs are detached if the current number exceeds the lowered TCBLIMIT.

Figure 5-3 shows the use of TCBs and threads during an 8 hour day life of the attachment:

- **+0 hours**
  - The attachment is started
  - TCBs are attached when work comes in requiring a thread to be created

- **+2 hours**
  - 30 total TCBs created by the attachment
  - 20 active threads as represented by the lower shaded area
  - 10 unused TCBs. These were used for (1) unprotected threads which are no longer in use, or (2) protected threads that have expired.

- **+4 hours**
  - 40 total TCBs, TCBLIMIT is reached. attachment will move unused TCBs between DB2ENTRYs as required

- **+**
  - End of peak use of threads
► +6 hours
  – Workload has dropped off, threads are terminated, most TCBs are unused. TCBLIMIT could be reduced.
► +8 hours
  – All TCBs are detached when the attachment is shut down.

Figure 5-3   Threads and TCBs

The maximum number of active threads is defined by the THREADLIMIT (equivalent of THRDA in RCT) parameter in DB2CONN (pool) and DB2ENTRY definition. There is no RDO equivalent of the THRDM parameter, so now there is no artificial limit on the value for THREADLIMIT.

Figure 5-4 shows same examples of THREADLIMIT setting:
► Example (a) has a THREADLIMIT of 2. For the first transaction using the DB2ENTRY, the attachment will attach a TCB for the thread required. A second transaction will cause a second TCB to be attached. If a third transaction comes in while the first two are busy, it will:
  – Wait for one of the threads to become free (THREADWAIT(YES))
  – Route the request to the pool (THREADWAIT(POOL))
  – Abend the transaction (THREADWAIT(NO))
► Example (b) has THREADLIMIT(3) allowing a maximum of three TCB and three threads
► Example (c) has THREADLIMIT(2) allowing a maximum of two TCBs and two threads
► Example (d) has THREADLIMIT(0) so all requests are routed to the pool. This DB2ENTRY will never have any of its own TCBs.
► Example (e) is the definition of the pool in the DB2CONN definition. It has specified a THREADLIMIT of 3.
Protected threads

Protected threads are indicated by PROTECTNUM on a DB2ENTRY (or THRDS>0 for an RCT entry) and are closely related to thread reuse. The pool does not have protected threads. A protected thread will remain available for reuse after released for 2 purge cycles. The PURGECYCLE (or TYPE=INIT PURGEC in RCT) parameter sets this value and the default is 30 seconds. Thus, the average time an unused thread will remain allocated is 1.5*PURGECYCLE. If the thread is unused after the end of the purge cycle, it will be terminated. The unprotected threads are indicated by THREADLIMIT - PROTECTNUM (or THRDA - THRDS in RCT). So for an entry, if THREADLIMIT=5 and PROTECTNUM=2, there would be 2 protected threads and 3 unprotected.

The drawback of protected threads is that the thread resources (memory, plan locks, etc.) remain held even when no transaction is using it. This can cause contention with other DB2 tasks, like utilities which require exclusive access or a rebind.

DB2ENTRY THREADWAIT

Specifying THREADWAIT(YES) means that the attachment will wait for SQL transactions when no threads are available. The use of CICS TRANCLASS may be a more efficient method of limiting the number of tasks for a particular DB2ENTRY. Set the MAXACTIVE value in TRANCLASS for the transaction to THREADLIMIT + 1. That will permit only one task to wait in the attachment and allow unprotected threads to be reused during high volume periods. Consider increasing the THREADLIMIT if more threads are needed.

THREADWAIT(POOL) is the recommended setting for most DB2ENTRYs. However be aware though that if high volume transactions are not limited via MAXACTIVE or another method, they could monopolize pool threads. Set MAXACTIVE in TRANCLASS to THREADLIMIT+n where n is the number of transactions allowed to overflow to the pool.

THREADWAIT(NO) will abend the transaction if no threads or TCBs are available.

Decide if MAXACTIVE should be greater than THREADLIMIT allowing the application to report the busy condition or MAXACTIVE set to THREADLIMIT to queue additional transactions.
**DB2ENTRY versus running in the pool**

Most transactions should run in the pool or be defined to divert to the pool (THREADWAIT(POOL)) for temporary high volume periods, thus limiting the number of TCBs and unused threads. DB2ENTRY definitions should only be used to:

- Override pool settings, like AUTH or ACCOUNTREC, or
- Define TCBs (threads) for a specific high volume transaction

To make transactions run in the pool you have three alternatives:

- No DB2ENTRY: all transactions run in the pool
- DB2ENTRY with THREADLIMIT(0): all transactions run in the pool
- DB2ENTRY THREADLIMIT(n) THREADWAIT(POOL): only transactions that exceed the THREADLIMIT(n) run in the pool.

The simplest setup for the CICS/DB2 Attach is to have a DB2CONN definition but no DB2ENTRYs or DB2TRANs. With this setup, command threads are available to DB2, and all other transactions run in the pool. The DB2CONN definition must define an appropriate catch-all plan or use a dynamic plan exit. CICS regions with few programs may be best defined as a single plan which is then hard-coded in the DB2CONN definition. CICS regions with many programs should consider using a dynamic plan exit that determines the plan name from the transactions attributes, such as the program name or transaction ID. Be aware that this approach reduces the possibility of reusing existing threads and monitoring, or accounting becomes very difficult.

### 5.2.2 Monitoring

This section presents some hints and tips in CICS and DB2 monitoring to help you analyze the CICS/DB2 attach behavior.

**-DISPLAY THREAD**

The -DISPLAY THREAD DB2 command displays current status information about DB2 threads. With CICS TS V1.2 and higher the format of the correlation id passed to DB2 has changed. The full 12 bytes is now used and is made up as follow:

- The first four bytes denote an entry, pool, or command thread:
  - ENTR denotes an entry thread
  - POOL denotes a pool thread
  - COMD denotes a command thread

- The second four bytes are the transaction ID (transid). Note that the correlation can only be changed when a sign-on occurs. Hence, if a transaction reuses a thread and reuses the sign-on (static AUTHID) then the transid quoted in the correlation id will not necessarily match the executing transid.

- The last four bytes is a number. This number is unique across all threads used by the attachment. Previously, the two byte number in the old style correlation id was only unique within an entry.

Figure 5-5 shows a sample of a -DISPLAY THREAD output. The description of the output is keyed to the letters “a” through “f” on the diagram:

- a. This is the control TCB. This TCB is not used for processing SQL requests.
- b. This is a pool connection (first four letters POOL) executing a command (transaction DSNCE).
- c. This is a pool connection that last ran transaction XP11 but the thread has terminated.
- d. This is an active entry connection (first four letters ENTR) running transaction XP05.
e. This is an active entry connection running transaction XP05 with remote activity.
f. This is an active command connection executing a command.

```
DSNV401I = DISPLAY THREAD REPORT FOLLOWS -
DSNV402I = ACTIVE THREADS -
NAME       ST   A  REQ ID     AUTHID   PLAN     ASID   TOKEN
a) CICSTS12 N    3   SYSADM      001B     0
b) CICSTS12 T   *    9 POOLDSNC0005 SYSADM      001B   14
c) CICSTS12 N    5  POOLXP110002 SYSADM      001B     0
d) CICSTS12 T   *    4 ENTRXP050001 SYSADM   TESTP05  001B   15
e) CICSTS12 TR    4 ENTRXP050003 SYSADM   TESTP05  001B   16
               V444-DB2NBT.LUND0.AA8007132465=16 ACCESSING DATA AT
               V446-SAN_JOSE
f) CICSTS12 T   *    3 COMDDSNC0004 SYSADM      001B   19
DISPLAY ACTIVE REPORT COMPLETE
DSN9022I = DSNVDT '^-DIS THD' NORMAL COMPLETION
```

Figure 5-5 -DISPLAY THREAD

For any line other than the control TCB with low requests counts, you should consider reducing the number of threads and possibly completely eliminating the DB2ENTRY if the number of threads is at or near 1.

See **DB2 UDB for OS/390 and z/OS Version 7 Messages and Codes**, GC26-9940 for more information on the output of the command -DISPLAY THREAD.

**CEMT INQUIRE TASK**

The CICS INQUIRE TASK command can show you when task is suspended in the CICS/DB2 attachment. In Figure 5-6 you can see a sample output of CEMT INQUIRE TASK. This information is more detailed in CICS TS V1.2 and higher. It is possible to determine why the task is suspended in the CICS/DB2 attachment according to the values of htype and hvalue:

- `htype=DB2`, `hvalue=LOT_ECB`: Task is waiting for DB2
- `htype=CDB2RDYQ`, `hvalue=name of DB2ENTRY or *POOL`: Task waiting for a thread on named DB2ENTRY or the pool
- `htype=CDB2TCB`, `hvalue=null`: TCBLIMIT has been reached. Task waiting for a TCB to become available.

Figure 5-6  CEMT INQUIRE TASK outputs

The same information shown on the CEMT panel is reported in a CICS system dump in the dispatcher domain summary section. See Figure 5-7.

Figure 5-7  CICS system dump

**DSNC DISP STAT**

Attachment statistics are displayed when CICS is shut down and through the DSNC DISP STAT command. Statistics are shown for each DB2ENTRY. The counters can be reset with the CICS "RESETNOW" command. From the output shown in Figure 5-8 you can determine:

- XC01
  - The AUTHS counter shows 1; this value is good since there is only 1 sign-on and no un-necessary authorization processing takes place.
  - There is a lot of contention for the single thread defined. If the W/P column is greater than zero, you know that there were fewer threads than indicated in the HIGH column. W/P indicates the number of times that all available threads for this entry were busy.
  - We can see that there were 12 units of work (ABORTS + 1-PHASE+ 2-PHASE), thus there were potentially only 12 times we needed to schedule a thread. But, we waited 11 times- nearly 100%! We had a maximum of 2 requests at a time (HIGH), so every request except the first one had to wait for DB2 (time waiting for a thread and time processing the requesting).
► XC02, XA81, XCD4, XA88
  – If these entries have any protected threads, the zero value in the CALLS column indicates that they have not been used yet and maybe should not be protected.
► POOL
  – The calls column is zero so no transactions have run in the pool. This may mean that there are too many entry threads.
► XA20
  – Since this entry shows W/P=0 it has a dedicated thread. Since the transaction ran only once, a better solution would be to specify THREADLIMIT(0) and THREADWAIT(POOL) or remove the entry completely if the pool definition would be sufficient.
► XP03
  – This entry has similar W/P problems as XC01 (transactions waited 50% of the time).

<table>
<thead>
<tr>
<th>TRAN</th>
<th>PLAN</th>
<th>CALLS</th>
<th>AUTHS</th>
<th>W/P</th>
<th>HIGH</th>
<th>ABORTS</th>
<th>1-PHASE</th>
<th>2-PHASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>POOL</td>
<td>POOL</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XC01</td>
<td>DSNXC01</td>
<td>22</td>
<td>1</td>
<td>11</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>XC02</td>
<td>DSNXC02</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XA81</td>
<td>DSNA81</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XCD4</td>
<td>DSNCED4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>XP03</td>
<td>DSNTP03</td>
<td>50</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>XA20</td>
<td>DSNTA20</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>XA88</td>
<td>*</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

DSNC020I THE DISPLAY COMMAND IS COMPLETE

Figure 5-8  DSNC DISP STAT output

See CICS Transaction Server for z/OS V2.2 CICS DB2 Guide, SC34-6014 for a complete description of the command DSNC DISPLAY and its output.

**Measuring thread reuse: DB2 side**

Using DB2 PM long accounting report, with detail by connection (for example, the “CICSPROD” region), plan name and thread you can obtain information about thread reuse. DB2 Accounting trace Class 1 and 2 must be started.

If ACCOUNTREC(UOW) of DB2ENTRY or DB2CONN, then the RESIGNON value is the thread reuse percentage. Otherwise, if ACCOUNTREC is not UOW, you can determine the approximate thread reuse for a given plan using the fields COMMITS and DEALLOCATION (threads used) and the following formula:

Approximate % thread reuse = ((COMMITS - THREADS USED)/COMMITS) x 100%

This is just an estimate, since some applications may execute multiple units of work under one thread. For example, non-terminal tasks, tasks using cursor WITH HOLD and tasks modifying special registers.

Figure 5-9 shows parts of a DB2 PM report with the fields used to get thread reuse information.
Measuring thread reuse: CICS side

In addition to the statistics available in the output of the command DSNC DISP STAT command, and the output to STATSQUEUE destination of the DB2CONN during the CICS/DB2 attachment shutdown, there are specific DB2 statistics available among the ones CICS collects. CICS writes statistics to a System Management Facilities (SMF) data set. The records are of SMF type 110, sub-type 002. DB2 statistics values can be recorded and reported on demand, or on normal CICS intervals like any other CICS statistics:

- EXEC CICS PERFORM STATISTICS RECORD DB2
- EXEC CICS SET STATISTICS for interval, end-of-day, requested, requested and reset, and unsolicited
- EXEC CICS COLLECT STATISTICS SET (stat_buf) DB2CONN DB2ENTRY(entryname)

The following DB2 statistics are available in the DFHSTUP program:

- Resource statistics:
  - Resource information: Details of various attribute settings of each DB2ENTRY. For example: Plan Name, Thread Wait
  - Request information: Details of how many requests of various types have been performed against each DB2ENTRY. For example, Thread Reuse, Call count.
  - Performance information: Details of thread information for each DB2ENTRY. For example: Protected Thread (Pthread) Limit, Pthread Current.

- Summary statistics:
  - Summary global statistics: A summary of global information of the CICS/DB2 attachment. For example: DB2 Sysid, Connection Name, Total Number of Pool Thread Reuses.
  - Summary resource statistics: A summary version of each of the three Resource Statistics (Resource Information, Request Information and Performance Information).

For more information on CICS statistics, monitoring and a complete description of the CICS/DB2 statistics reports and fields, see CICS TS for z/OS V2.2 Performance Guide, SC34-6009.
Figure 5-10 shows sample JCL to invoke DFHSTUP utility selecting only DB2 resource type. The two main reports generated by this JCL are the Request, shown on Figure 5-11, and the Performance report, shown on Figure 5-12.

```
//DFHSTUP  JOB ,'CICS STATS',CLASS=A,
//*******************************************************
//STATS   EXEC PGM=DFHSTUP,REGION=7000K
//STEPLIB DD DISP=SHR,DSN=CICSVS.TS13.CICS.SDFHLOAD
//SORTWK01 DD UNIT=SYSDA,SPACE=(CYL,(2))
//SORTWK02 DD UNIT=SYSDA,SPACE=(CYL,(2))
//SORTWK03 DD UNIT=SYSDA,SPACE=(CYL,(2))
//SORTWK04 DD UNIT=SYSDA,SPACE=(CYL,(2))
//SORTWK05 DD UNIT=SYSDA,SPACE=(CYL,(2))
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=*  
//DFHSTATS DD DSN=SMF.STPLEX4A.D010910.T193358,DISP=SHR
//DFHSTWRK DD UNIT=SYSDA,SPACE=(CYL,(8,3))
//DFHPRINT DD SYSOUT=*  
//TRACEOUT DD DUMMY
//SYSUDUMP DD SYSOUT=*  
//SYSABEND DD SYSOUT=*  
//SYSIN DD *
SELECT TYPE=(DB2)  
COLLECTION TYPE=ALL
SUMMARY

Table 5-15 Sample Resource Statistics

<table>
<thead>
<tr>
<th>DB2Entry</th>
<th>Call</th>
<th>Signon</th>
<th>Commit</th>
<th>Abort</th>
<th>Single</th>
<th>Thread Reuse</th>
<th>Threads</th>
<th>Wait</th>
<th>Overflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRCI</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WRCB</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WRCF</td>
<td>770</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>72</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>24</td>
<td>23</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>286</td>
<td>43</td>
<td>0</td>
<td>0</td>
<td>43</td>
<td>41</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>251</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>25</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>55</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>89</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>30</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>322</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>23</td>
<td>17</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>WREDI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td><strong>TOTALS</strong></td>
<td>4478</td>
<td>146</td>
<td>0</td>
<td>0</td>
<td>146</td>
<td>123</td>
<td>18</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
```

The key DB2 statistics in the Request report are:

- **Thread reuse:** The value of thread reuse reported by DFHSTUP is always accurate, no matter the value of ACCOUNTREC in DB2ENTRY or DB2CONN.
Thread Wait: The field Thread Wait/Overflow means the number of times all available threads in the pool were busy and a transaction had to wait for a thread to become available. This count includes transactions that overflow to the pool to acquire a thread and have to wait for a pool thread. The Thread Wait/Overflows tells you only how many times a task waits for a thread, and not how long.

CPU accounting

Information about CPU accounting in DB2 is collected by activating DB2 Accounting trace Class 1 and Class 2. Class 1 results in accounting data being accumulated by several DB2 components. The elapsed time of a DB2 thread is included in this data. Class 2 collects the elapsed and processor times spent in DB2.

Figure 5-13 shows the flow of where CPU is used when processing a transaction. Class 1 CPU is the sum of CPU used in segments 2, 3, 4, 6, 7, and 8 and Class 2 CPU is the sum of segments 3 and 7. Segments 1, 5, and 9 are in the CICS main TCB, so that CPU is not included in the DB2 Class 1 CPU time.
Most of the tuning presented here affects segments 2, 4, 6 and 8. So, as parameters are adjusted, a DB2 PM accounting report can be used to see if the conditions have improved or worsened. You can estimate the CPU utilization of the attachment overhead with the following formula:

\[(DB2 \text{ Class 1 CPU}) - (DB2 \text{ Class 2 CPU})\]

Usually, this overhead is very small and in most situations where we have seen larger values, the RACF sign-on exit was in effect.

Some of the attachment activity CPU usage is recorded in the segments 1, 5, and 9, such as dynamic plan selection, thread assignment, so adjustments to those areas are not reflected in the above formula.

Class 1 and Class 2 elapsed time should not be used because it can include time when a thread is idle and waiting for work. This is mainly true for protected threads since an unprotected thread will be deallocated if no task is waiting.

Using the CICS Performance Analyzer to analyze the CICS performance class records you can check the elapsed time a transaction spends waiting for a DB2 request. Also check the attach overhead by the CICS side. The fields used are:

- \text{RMISUSP}: The total elapsed time the task was suspended by the CICS dispatcher while in the CICS Resource Manager Interface (RMI).
- \text{DISPWAIT}: The time task waited to resume execution.
- \text{RMITIME}: Amount of elapsed time spent in the Resource Manager Interface (RMI).

The elapsed time a transaction spends waiting for a DB2 request can be calculated as:

\[\text{RMISUSP} - \text{DISPWAIT}\]

The attach overhead can be calculated as:

\[\text{RMITIME} - \text{RMISUSP} - \text{DISPWAIT}\]

Also, in the CICS performance class, you have DB2 statistics that can be very useful:

- \text{DB2CONWT}: The elapsed time in which the user task waited for a CICS/DB2 subtask to become available.
- \text{DB2RDYQW}: The elapsed time in which the user task waited for a DB2 thread to become available.
- \text{DB2REQCT}: The total number of DB2 EXEC SQL and Instrumentation Facility Interface (IFI) requests issued by the user task.
- \text{DB2WAIT}: The elapsed time in which the user task waited for DB2 to service the DB2 EXEC SQL and IFI requests issued by the user task.

### 5.2.3 Performance and tuning considerations

Figure 5-14 shows the activities that add up to the user’s perceived response time. The real goal is to reduce response time for some or all work on a machine. When attempting to tune the attachment, one needs to be aware of what benefits we can get in relation to the workload. If the transactions running are longer query type, the relative benefits of tuning will be less. When we are tuning the attachment, we are trying to minimize:

- The attachment overhead
- The thread sign-on costs
- The thread creation cost
Thread reuse

A task with SQL requests can obtain a thread to connect to DB2 by two ways:

- Reuse an existing thread that another task used previously or
- Allocate a new thread

The most efficient way for a transaction to execute an SQL statement is to reuse a thread rather than allocate a new one. A thread can be reused only if the DB2ENTRY associated with the transaction is the same as the previous transaction and the new transaction uses the same plan.

Also, to reuse a thread, the DB2 thread definition (DB2ENTRY or DB2CONN for pool entries) must have either:

- THREADWAIT=YES to allow the transaction to wait for a thread already in use or
- PROTECTNUM > 0 (protected thread) to allow unused threads to wait for a transaction that might otherwise run immediately if allowed to create a new thread.

Figure 5-15 shows a DB2ENTRY screen fragment with this options.
A thread becomes available for reuse after SYNCPONTS for terminal driven transactions if the thread is in the initial state. Initial state means that all modifiable special registers are at their initial value and there are no held cursors. For non-terminal driven transactions prior to CICS TS V1.2, threads are only released at end of task (EOT). With CICS TS V1.2 or higher you can set option NONTERMREL(YES) on the DB2CONN definition to cause non-terminal driven transactions to release their threads at SYNCPONNT, subject to the same rules for terminal driven.

The modifiable special registers include:

- CURRENT APPLICATION ENCODING SCHEME
- CURRENT DEGREE
- CURRENT LOCALE LCCTYPE
- CURRENT OPTIMIZATION HINT
- CURRENT PACKAGESET
- CURRENT PATH
- CURRENT PRECISION
- CURRENT RULES
- CURRENT SERVER
- CURRENT SQLID

Unprotected threads are terminated immediately upon thread release if no other task is waiting. This means that a task that releases its thread at an intermediate SYNCPONNT may have to go through thread creation multiple times within one transaction. Note that unprotected threads can be reused. It is a common fallacy that you must use a protected thread to get thread reuse. Thus, pool threads can be reused, though less likely if many different plans are executing in the pool.

The advantages of reusing threads is more significant in short and high volume transactions. In a short transaction, the CPU savings of avoiding the creation of a new thread is significant while they are less significant for long transactions. Thread deallocation is asynchronous and does not take as much CPU so it is less of a consideration when tuning.

**Sign-on**

Figure 5-16 shows the thread protocol sequence. The three main states are:

- The IDENTIFY state of a thread indicates that the TCB is known to DB2. This shows as status N in a DISPLAY THREAD command.
- The SIGN-ON state indicates that DB2 has processed and approved the authorization ID for the thread against the plan name. A TCB in this state will have a value in the AUTHID column of the DISPLAY THREAD command
- The CREATED state indicates that DB2 has allocated the plan and can process SQL requests. The presence of a plan name in the PLAN column and a status of T of DISPLAY THREAD command indicates this state. A CREATED thread may be signed on again without being recreated. To recreate a thread, it must first be terminated.

Though less significant than thread reuse, sign-on reuse may provide valuable performance gains for very short transactions (for instance, a single SQL statement executed in the task).
Sign-on always occurs when a thread is used for the first time. A thread is signed on again when the thread is reused and any of the following occur:

- The primary authorization ID changes: The AUTHID or AUTHTYPE parameter value will greatly affect the frequency that the primary authorization changes
  - Best choice is a value that is constant such as a AUTHID(string) or AUTHTYPE(SIGN).
  - the worst choice is a value that may change frequently such as AUTHTYPE(TERM) or AUTHTYPE(USERID). Note a thread can be reused but a re-sign-on will occur if the authorization values of the thread have to be changed.
  - Prior to CICS TS V1.2, note that three values on AUTH=(x1,x2,x3) are tried in order until a value is available for use as primary authorization id. The attachment does not use all three values as a collective authorization list.
- The first SQL statement after a SYNCPOINT if ACCOUNTREC(UOW) (TOKENE=YES in RCT) is specified. This provides the ability to correlate DB2 accounting with CICS accounting on a per UOW transaction basis. For transactions with multiple UOWs per transactions, multiple DB2 accounting records have to be correlated. With CICS TS V1.2 this can be overcome by specifying ACCOUNTREC(TASK).
- The TXID changes on a DB2ENTRY used by multiple transactions. This can be disabled prior to CICS TS V1.2 by specifying TXIDSO=NO. With CICS TS V1.2 and higher, the same behavior occurs when ACCOUNTREC(TXID) is specified and can be avoided by using ACCOUNTREC(Task) or ACCOUNTREC(None)
- The last transaction left an open held cursor or left a special register in a modified state.
If you do not use DB2 security and set grant access to PUBLIC, you should also specify CACHESIZE(0) for the BIND PLAN.

Avoiding a sign-on wherever possible will improve the performance of transactions within a DB2ENTRY.

**DB2 Bind: more reuse**

A transaction can save the cost of DB2 resource allocation by specifying ACQUIRE(ALLOCATE) and RELEASE(DEALLOCATE) in the BIND options. In a DB2 data sharing environment this provides significant reduction in messages to the coupling facility. The disadvantages of this option are that locking concurrency problems can occur if the resources remain allocated for a long time, and the EDMPOOL will grow as more packages are used within the life of a thread. For more information about bind options, see “Coordinating DB2CONN, DB2ENTRY, and BIND options” in CICS TS for z/OS V2.1 CICS DB2 Guide, SC34-5707.

**Avoid the queue at create thread**

If a transaction comes in and no thread can be reused, one is created if all of the following occur:

- The number of threads for the entry (or pool) is less than THREADLIMIT
- The number of TCBs is less than TCBLIMIT
- The total number of threads in the DB2 subsystem is less than the CTHREAD setting in DSNZPARM

If TCBLIMIT is exceeded, there is no TCB detaching mechanism. The CICS/DB2 attachment will move TCBs between entries as required. If all TCBs are in use, then the THREADWAIT parameter will be used to determine whether to wait for a TCB to become available, or abend the transaction. You can use:

- CEMT INQUIRE DB2CONN to see how many TCBs have been attached
- CEMT SET DB2CONN TCBLIMIT to alter the number of TCBs
- CEMT SET DB2CONN THREADLIMIT to alter the number of pool threads
- CEMT SET DB2ENTRY THREADLIMIT to alter the number of threads of a particular DB2ENTRY
To check TCBLIMIT or THREADLIMIT is being exceeded you can use:

- CICS/DB2 statistics will report on current and peak number of threads used per entry, and current and peak number of TCBs attached and the number of thread reuses.

- CEMT INQUIRE TASK will show:
  - htype(CDB2RDYQ) and hvalue(entry name) or ‘pool’ if the transaction is waiting for a thread
  - htype(CDB2TCB) if the transaction is waiting for a TCB

To determine if CTHREAD has been reached you can use:

- -DISPLAY THREAD command
- DB2 PM Statistics report

**Good application design**

There are some guidelines in coding applications to make threads reusable:

- Close cursors when no longer needed
- Restore special registers to their initial state
- Issue SYNCPOINT as soon as possible

When a thread has open held cursors or modified special registers, the thread cannot be reused at intermediate SYNCPOINTS (before end of task (EOT)). A partial sign-on will occur at EOT to restore the thread to the initial state. That is why is a good practice to close all cursors, especially the ones with the WITH HOLD option, when they are no longer needed to release locks and make the thread reusable. The extra effort it takes to restore the special register to the initial value may well be worth the performance increase achieved by making the thread reusable.
Also, there are some considerations regarding the use of plan and packages:

- Use a single large plan with many packages rather than using Dynamic Plan Switching (DSP), PLANEXITNAME(nnn) in DB2 ENTRY to switch between many smaller plans. Using DSP reduces the possibility of reusing an existing thread. Also, a transaction can only switch plans when the thread is reusable.

- Package definitions should also be tuned. List the most frequently used packages first and use wildcarding where possible. A large collection does not present a significant performance impact.

**DB2ENTRY PRIORITY**
The PRIORITY(DPMODE in RCT) parameter of DB2CONN and DB2ENTRY sets the dispatching priority of the thread TCB relative to the CICS main TCB. The options are HIGH, EQUAL and LOW. The default value is HIGH. *You should not change this value unless you are nearing 100% CPU utilization (>90%) for your S/390.* There are some efficiencies in keeping CICS busy so that the QR TCB does not suspend waiting for work. Changes to PRIORITY will have little effect when the CPU usage is less than 100%.

If you need to slow down the DB2 processing to provide more CPU for CICS, here are some rules of thumb to pick the settings:

- HIGH should be used for transactions that are high volume and many simple SQL statements. Thus, when the CICS main task wakes up there will be multiple tasks with completed SQL ready for execution.
  - Note that HIGH can starve the CICS region of CPU resources if too many transactions run at one time.
  - Use a weighted approach to determine which transactions may be the best candidates (#SQL statements per transaction) * (frequency of transaction)

- EQUAL should be used for most other transactions, especially those that have other non-DB2 activity (such as access of VSAM files, IMS, etc.)

- LOW is very similar to EQUAL. Both EQUAL and LOW are below the CICS main TCB dispatch priority. LOW is usually used for utility type transactions and EQUAL for long running query type transactions.

Changing the DB2address spaces priorities may adversely affect CICS throughput. Most work is done in the allied address space (such as a CICS address space TCB).

### 5.2.4 Conclusion

The most important improvement in performance and in resource usage presented here is the thread reuse approach. It will be more significant for high volume and short transactions. Avoiding sign-ons, although it is not as significant as thread reuse, can save you some resource utilization and give you some performance gain.

### 5.3 CICS/DB2 attachment in CICS TS V2.2

In this section we provide a brief description of the enhancements introduced by CICS TS V2.2 to the CICS/DB2 interface. See *CICS TS V2.2 Release Guide*, GC34-5983 and *CICS TS V2.2 CICS DB2 Guide*, SC34-6014 for further information.
5.3.1 OTE

The CICS/DB2 attachment facility now exploits the open transaction environment (OTE) to enable the CICS/DB2 task-related user exit to invoke and return from DB2 without switching TCBs. To gain the performance benefits of the OTE, CICS must be connected to DB2 version 6 or later, and uses a threadsafe application program.

The CICS OTE was introduced to enable applications to perform, under an open TCB, actions that are not permitted under the CICS quasi-reentrant (QR) TCB. This is because CICS runs user transactions under a single OS/390 TCB, the QR TCB, and direct invocation of other services outside the scope of CICS permitted interfaces could interfere with CICS own use of this TCB. In particular, services that result in the suspension (blocking) of the QR TCB would cause all CICS tasks to wait.

Initially, OTE was exploited by only Java applications that run in a Java Virtual Machine (JVM), enabling each JVM to run under its own TCB. Because no other task shares the JVM's TCB, JVM programs can safely use non-CICS application programming interfaces, and avoid impacting other user tasks that run under the CICS QR TCB. Later, support to enable hot-pooled high performance java (HPJ) programs to exploit OTE was added.

OTE support is provided mainly through pools of open TCBs of different modes, which enable designated tasks to obtain a suitable TCB for the lifetime of the task. With the new enhancements to OTE, the open TCB modes are extended to enable task-related user exits to exploit OTE and avoid the need to manage a private pool of TCBs. The CICS/DB2 adaptor is the first task-related user exit to exploit this OTE enhancement.

CICS TS V2.2 has three separate pools of open TCBs:
- H8: TCBs allocated by hot-pool HPJ-compiled Java program.
- J8: TCBs allocated for execution of a JVM program (Java programs that require a JVM).
- L8: TCBs allocated for non-Java program accessing a resource manager through a task-related user exit enabled with OPENAPI option. Used by the CICS/DB2 attachment.

The OPENAPI option specifies that the task-related user exit program is using non-CICS APIs.

If the user application program that invokes the task-related user exit is defined as quasi-reentrant, CICS switches the user task to an open TCB before passing control to the task-related user exit program. CICS assumes that a task-related user exit, enabled with OPENAPI, does not manage its own private pool of TCBs for non-CICS services. Conversely, if OPENAPI is omitted, CICS will assume that the task-related user exit is either using only the CICS API, or that it performs its own TCB switch to invoke non-CICS services.

Threadsafe

In a open transaction environment, programs that access shared resources must be aware that these resources can also be accessed by other user task running in a open TCB. Programs that use appropriate serialization techniques when accessing shared resources are described as threadsafe. It is defined with the attribute CONCURRENCY(THREADSAFE) on the program definition. For most resources, such as files, transient data queues, temporary storage queues, and DB2 tables, CICS processing automatically ensures access in a threadsafe manner. However, for any other resources, such as shared storage, which are accessed directly by user programs, it is the responsibility of the user program to ensure threadsafe processing. Typical examples of shared storage are the CICS CWA, global user exit global work areas, and storage acquired by EXEC CICS GETMAIN SHARED commands.

For further details see CICS Application Programming Guide, SC34-5702.
CICS/DB2 attachment with and without OTE

When a CICS application running under the CICS QR TCB invokes DB2 it is necessary for the CICS/DB2 adapter to switch control to a different TCB because the DB2 invocation may involve a wait for I/O. Hence, control is switched to a different TCB so that waiting for I/O will only hold up the CICS transaction that issued the DB2 request. There has to be a another TCB switch on return from DB2. These TCB switches are relatively expensive in CPU cost. This can be seen in the left side of the Figure 5-19.

![Figure 5-19 Pre-OTE and OTE CICS/DB2 attach](image)

If the CICS is connected to DB2 V6 or later, the task-related user exit is enabled with OPENAPI automatically for you when you start the CICS/DB2 adaptor. So, if the CICS program is threadsafe, then it can run under its own TCB (an L8 TCB), the same one that executes the DB2 commands as shown on the right in figure Figure 5-19. The transaction starts on the QR TCB, but when DB2 is invoked, CICS switches control to an L8 TCB dedicated to this transaction. On return from DB2, if the application is threadsafe, there is no need to return to QR TCB. It continues to executes in the L8 TCB.

Similarly, there is no need to switch TCBs to call CICS commands that are threadsafe. But, it is necessary to switch back to the QR TCB to perform some CICS commands and other functions, such as the syncpoint process. Usually, depending on the number of SQL calls and how they are interleaved with non-threadsafe EXEC CICS commands, the number of TCB switches will be reduced. Note that the CICS application program must be threadsafe for this function to be exploited, because two instances of the program may be executing concurrently. If the application program is not threadsafe, then it will operate like it was before.
Some EXEC CICS commands are not threadsafe. However a PROGRAM using non-threadsafe COMMANDS could still be threadsafe because CICS switches control to the QR TCB when such a command is executed. However, execution of a non-threadsafe COMMAND will cause a TCB switch which will reduce the effectiveness of this performance enhancement.

OTE exploitation is important for enterprise beans that make DB2 requests. If the CICS/DB2 task-related user exit operates as quasi-reentrant, and does not exploit the open transaction environment, four TCB switches are needed for each DB2 request made by an enterprise bean. There is a switch from the enterprise bean's TCB to the CICS QR TCB, where the CICS/DB2 task-related user exit is invoked. Then, there is a switch to one of the TCBs managed by the task-related user exit, where the DB2 subtask is carried out.

Another switch is needed to return to the task-related user exit on the CICS QR TCB, and a final switch to return to the enterprise bean on its TCB. If the CICS/DB2 task-related user exit is threadsafe and with open API, and exploits the open transaction environment, only two TCB switches are needed for each DB2 request: from the enterprise bean's TCB to the L8 TCB, where the task-related user exit is invoked, and where the DB2 requests are made, and then back to the enterprise bean's TCB.

**Monitoring**
The switch of CPU consumption from the DB2 TCB to the OTE TCB implies a change in the way that CPU time appears in the SMF monitoring records.

**Performance measurements**
The performance measurements were made using the IBM Relational Warehouse Workload (IRWW). The environment used in the IRWW is:

- LPAR in a G6 machine
- z/OS V1.2
- CICS TS V2.2
- CICS TS V1.3
- DB2 for z/OS and OS/390 V7

CICS TS V1.3 was used as baseline for the measurements, since CICS TS V2.1 was announced with a limited duration of program services. The throughput measurements shows the performance gain when CICS TS V2.2 is connected to DB2 V7, and therefore, exploits the use of OTE for the CICS/DB2 attachment, together with threadsafe applications.

Running a transaction with one million fetches of a single character column from a 2 million row table demonstrates the maximum potential CPU savings of eliminating the task switch between the QR TCB and the thread TCB.

Figure 5-20 compares the most efficient environment on TS 1.3, CONCURRENCY(QuasiRent) and on TS 2.2 CONCURRENCY(THREADSAFE).

These measurements were done with CICS TS 2.2 and CICS TS 1.3 using DB2 7.1.
CICS TS 1.3 was used as the base because it is the last release that manages the thread TCBs directly under the attachment code. Later releases use the L8 TCBs, but they do not support application code running under the same TCB as the DB2 thread (that is, until CICS TS 2.2.).

DB2 V7 was chosen because it is the latest release of DB2 currently available. DB2 V6 is the minimum release which supports this new feature.
The CPU time was taken from the job step CPU of the CICS region job output. CRLP was used to make the job completely automatic.

An initial baseline measurement was taken which executed no transactions to determine the CPU for starting and stopping the relevant CICS region (CICS TS 1.3 or 2.2). Next the same job was modified to include the 1 million row fetch transaction. Subtracting the CPU from step 1 above gives us the base cost for the transaction alone (no CICS startup or shutdown costs).

The delta between CICS TS 1.3 and CICS TS 2.2 of step 2 then gives us the cost savings between CICS TS 1.3 and CICS TS 2.2 running a transaction which fetches 1 million rows.

The CPU cost savings is $(1 - 29.1/52.1)^*100 = 44\%$.

![OTE cost savings: CICS TS 1.3 vs TS 2.2](Figure 5-20 Cost savings with OTE)

Transactions which connect to DB2 from CICS TS 2.2 and above will reflect a change in collected DB2 class 1 accounting information. This is not a change in how DB2 collects the information but, rather, a change in how CICS processes SQL requests. The thread TCB is the TCB which CICS uses to process SQL requests. Prior to CICS TS 2.2, very little activity other than the SQL request itself was executed on the thread TCB. CICS TS 2.2 will process much of the application, CICS API calls, CICS tracing, and other activity on the thread TCB. This change results in much higher class 1 CPU times reflected in DB2 SMF 101 records. This CPU time previously showed up in other CICS TCBs such as QR.

**Conclusions**

This is the absolute best case. In real life, your improvement will vary, depending on the SQL statements.
5.3.2 Group attach

With DB2 V7 and CICS TS V2.2, the DB2 group attach support allows a CICS system to connect to DB2 by specifying the name of a DB2 data sharing group instead of a specific DB2 subsystem. Connection will be made to one of the active members of the group residing on the same MVS image as CICS members that are active on other MVS images are not eligible for selection.

To activate the group attach facility, use the new DB2GROUPID attribute of the DB2CONN definition to specify the ID for the data sharing group instead of use the DB2ID attribute to specify a DB2 subsystem. Use the CEMT or EXEC CICS INQUIRE DB2CONN DB2ID() command when the connection has been established to find out which member of the data sharing group has been chosen for the current connection. This function is particularly useful when cloning application owning regions (AORs) interacting with a DB2 data sharing group.

Group attach and in-doubt resolution of units of work

Using group attach with CICS raises the question about in-doubt units of work (UOWs). If CICS is connected to member 1 of a data sharing group and the connection is lost, in-doubt UOWs may be held by this member. If CICS reconnects to member 1, these in-doubt UOWs can be resolved. If CICS connects to other members, these in-doubt UOWs cannot be resolved.

To solve this problem, CICS maintains a history of the last DB2 data sharing group member to which it was connected, and checks to see if there are in-doubt UOWs to be resolved. If there are any, CICS will attempt to connect to that member instead of using the group attach.

You can bypass this by specifying RESYNCMEMBER=NO in DB2CONN. DB2 will try one attempt to reconnect to the last data sharing group member. If the connection is successful, the in-doubt UOWs can be resolved, otherwise CICS uses group attach to connect to any DB2 data sharing member. The message DFHDB2064 is issued stating that there may be unresolved in-doubt UOWs with the latest recorded member.

5.3.3 Usability enhancement

This minor enhancement to the usability of the master terminal CEMT INQUIRE DB2TRAN command allows you to display the PLAN directly, avoiding the need to determine the DB2ENTRY first. In earlier releases, you have to make two separate inquires to find this information, because the transaction ID is part of the DB2TRAN definition. The plan name, or plan exit name, is part of the DB2ENTRY definition.

For example, when you use this command:

```cemt
CEMT INQUIRE DB2TRAN(*) TRANSID(ABCD)
```

The inquire returns the name of the DB2TRAN with which transaction ABCD is associated. Also the name of the plan or dynamic plan exit that transaction ABCD uses, is returned.

If you issue this command:

```cemt
CEMT INQUIRE DB2TRAN(*) PLAN(WXYZ)
```

This returns the names of all the DB2TRANs that use plan WXYZ, and the IDs of the transactions that are associated with those DB2TRAN definitions.
5.3.4 RMI PURGE enhancement

The new Resource Manager Interface (RMI) purge option is introduced in CICS TS V2.2. The option allows the writer of a task-related user exit (TRUE) to specify whether, before calling it, the RMI should defer purge and deactivate runaway. This offers an operational enhancement by making it easier to recover from system stall situations in CICS. The CICS/DB2 Attachment Facility uses the new RMI purge option when CICS is connected to DB2 V6 or higher. This allows applications waiting in DB2 to be purged from CICS. Previously, only forcepurge, with its danger of loss of integrity, was supported.

5.3.5 Other DB2 complementary functions

- Java SDK 1.3 J2EE
- Integrated CICS Translator COBOL coprocessor
- TCP/IP including ECI
Real time statistics

In this chapter, we describe the new function, real time statistics (RTS), added to DB2 V7 after GA. This function provides the necessary statistics that end users or automated task schedulers can use to determine which objects require REORG, RUNSTATS or COPY. Details are provided in the following sections:

- Description
- How real time statistics work
- Enabling RTS on your system
- Operation considerations
- DSNACCOR stored procedure
- Performance considerations
6.1 Description

Real time statistics enhancement is introduced by APARs PQ48447, PQ48448, PQ46859, and PQ56256. With RTS, DB2 provides the necessary information that end users or automated task schedulers can use to determine which objects require REORG, RUNSTATS or COPY. This will reduce the overall cost of running a DB2 environment. Some of the statistics collected are:

- The number of rows, LOB values or index entries modified since last REORG, RUNSTATS or COPY
- The physical space information such as the number of pre-formatted pages, allocated space, and the number of extents
- The number of distinct pages updates and the time of the first update since the last COPY

DB2 *always* generates real time statistics in memory for each table space and index space on your system. Statistics are generated for each partition for partitioned table spaces, and indexes. Optionally, these in-memory statistics can be externalized to DB2 tables from time to time, or when necessary. You can decide when to run REORG, RUNSTATS and COPY utilities, or when to enlarge your data sets by querying these tables. A new sample stored procedure, DSNACCOR, is provided to help you in this task.

One usage of the statistics tables is by the Control Center 390 (CC/390). CC/390 queries the DB2 catalog and statistics tables, and identifies the pages sets that require REORG, RUNSTATS or COPY. CC/390 uses the default criteria, or user specified criteria, to identify the page sets. CC/390 also generates utility statements to perform the utility tasks.

You can find further information about RTS in *DB2 UDB for OS/390 and z/OS Version 7 Administration Guide*, SC26-9931-01, in Appendix G and H. You can get the latest version of this standard DB2 manual from the following link:


6.2 Enabling RTS on your system

You must implement the following steps in order to externalize the in-memory statistics to DB2 tables:

- Create real time statistics objects
- Set interval for writing statistics
- Start the real time statistics database

DB2 uses a block of 152 bytes to store the statistics for each page set. So, the amount of storage needed to keep the in-memory statistics is:

\[
\text{amount of storage in bytes} = \text{maximum concurrent objects updated} \times 152 \text{ bytes}
\]

You should estimate the peak number of objects that might be updated concurrently. Keep in mind that the storage will only be freed after DB2 processes the statistics. This storage is allocated in the DBM1 address space. Because you cannot prevent DB2 from collecting the RTS in memory, if you are already facing DBM1 storage problems, monitor DBM1 storage usage before and after applying PQ48448, and make the necessarily adjustments.
Migration, fallback, and coexistence considerations

After the first migration to DB2 V7, you must create the statistics objects that are needed to contain the statistics. Subsequent migrations to V7 require the statistics tables to be emptied (SQL DELETE) to create accurate statistics.

After a fallback from DB2 V7, the statistics in DSNRTSDB.DSNRTSTS are no longer maintained and thus become down level.

In a data sharing coexistence environment, the statistics can be inaccurate until all DB2 members are updated to DB2 V7.

6.3 How real time statistics work

Real time statistics in-memory are always generated by DB2, kept in memory, and externalized when necessary. To externalize the statistics, DB2 examines the in-memory statistics, calculates the new totals, updates the new real time statistics tables with the new totals, and resets the in-memory statistics. This process is an asynchronous task. Utilities have an effect upon the statistics, but the changes are synchronous to the utility operations.

When externalizing in-memory statistics, DB2 inserts a row for each partition or non-partitioned page set in SYSIBM.TABLESPACESTATS or SYSIBM.INDEXSPACESTATS. If a row exists, it will be updated. The absolute statistic values (for example, TotalRows) are replaced with the new values, and incremental values are summed with the in-memory statistics. If for some reason, like lock contention or other resource unavailable, DB2 cannot update the statistics tables, this will not cause the requester to fail. A message (DSNI037I) will be written to the console and a new attempt to externalization will be made in the next update cycle.

The in-memory statistics control block for a page set is usually allocated when it is first created and marked as logically deleted at close. This storage is freed by DB2 after processing the statistics.

SQL INSERT, UPDATE, DELETE, and ROLLBACK statements, and certain utilities cause the in-memory real time statistics to be changed. This will be discussed later in 6.4.2, “When real time statistics are updated” on page 91.

6.3.1 Create real time statistics objects

You must create the following objects in order to allow DB2 externalizes the in-memory statistics:

<table>
<thead>
<tr>
<th>Table 6-1  RTS objects</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object name</strong></td>
</tr>
<tr>
<td>DSNRTSDB</td>
</tr>
<tr>
<td>DSNRTSTS</td>
</tr>
<tr>
<td>SYSIBM.TABLESPACESTATS</td>
</tr>
<tr>
<td>SYSIBM.INDEXSPACESTATS</td>
</tr>
</tbody>
</table>
Sample SQL statements to create these objects are available in member DSNTESS of DSN710.SDSNSAMP. These objects are a user created database and table space, but the name of each object, the column names and attributes, the CCSID EBCDIC, and LOCKSIZE ROW should not be changed. LOCKMAX 0 is recommended. Each table requires its corresponding index to be created for DB2’s usage. However, additional indexes can be create on tables SYSIBM.TABLESPACESTATS and SYSIBM.INDEXSPACESTATS. You need the proper authority to create these objects.

You can segregate these tables and indexes to a specific buffer pool for better performance, and ease of monitoring. When the statistics table pages/indexes are in the buffer pool, the speed at which in-memory statistics are written to the tables improves.

Before you create, alter, or drop an object in the statistics database, you must stop it first.

### 6.3.2 Set interval for writing statistics

DB2 considers writing the in-memory statistics to real time statistics tables based upon a user specified time interval. This value is changed through a system parameter.

You set the interval for writing real time statistics on field REAL TIME STATS on panel DSNTIPO of the installation CLIST. It can also be updated by dynamically modifying the system parameter STATSINST. The default value is 30 minutes and the value must be between 1 and 65535.

### 6.3.3 Start the real time statistics database

After you create the necessary objects, you must explicitly issue the command START DATABASE(DSNRTSDB) and ensure it is started in read-write mode to make it possible for DB2 to externalize real time statistics.

After you create the RTS database, DB2 immediately puts it into a stopped state. Stopping the database will ensure that the database is explicitly started by the start database command to enable the externalization of the real time statistics. Starting database DSNRTSDB is an implicit request for DB2 to write the in-memory statistics to the real time statistic table. During this start database process, the statistics objects are validated and if they are correct, DB2 enables the process to externalize in-memory statistics.

### 6.4 Operation considerations

You need to know when DB2 collects and externalizes the real time statistics, and what factors in your system can affect them in order to use them effectively.

#### 6.4.1 When real time statistics are externalized

Table 6-2 shows the when in-memory statistics are or are not externalized.
Table 6-2  When real time statistics are externalized

<table>
<thead>
<tr>
<th>When</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>At the end of STATSINT interval</td>
<td>All in-memory statistics are written to tables.</td>
</tr>
<tr>
<td>-STOP DATABASE(DSNRTSDB)</td>
<td>This command externalizes statistics for all objects in the subsystem and stops the real time statistics writing process.</td>
</tr>
<tr>
<td>-STOP DATABASE(db-name) SPACENAM(space-name)</td>
<td>This command externalizes statistics only for db-name and space-name</td>
</tr>
<tr>
<td>-STOP DB2 MODE(QUIESCE)</td>
<td>All in-memory statistics are written to tables.</td>
</tr>
<tr>
<td>-STOP DB2 MODE(FORCE)</td>
<td>No in-memory statistics are written, then you lose them</td>
</tr>
<tr>
<td>RUNSTATS UPDATE ALL</td>
<td>All in-memory statistics are externalized when the RUNSTATS jobs starts.</td>
</tr>
<tr>
<td>COPY</td>
<td>All in-memory statistics are externalized</td>
</tr>
<tr>
<td>Utility with any of the RTS objects in the utility list</td>
<td>No statistics will be externalized for any of the RTS objects in the utility list</td>
</tr>
<tr>
<td>Work files database and TEMP database</td>
<td>Only NACTIVE, SPACE, and EXTENTS statistics are collected and externalized</td>
</tr>
<tr>
<td>Read-only objects</td>
<td>No statistics are externalized</td>
</tr>
<tr>
<td>Tracker site</td>
<td>No statistics are externalized in a tracker site</td>
</tr>
</tbody>
</table>

Once you have enabled RTS in an existing DB2 subsystem, for newly created table spaces and indexes, rows are inserted into RTS tables at CREATE time, and:

- The column LOADRLASTTIME is set to CREATE timestamp
- The columns REORGLASTTIME, STATSLASTTIME, COPYLASTTIME are set to null

For table spaces and indexes that existed before RTS is enabled, rows are inserted when the objects are first updated at the next STATSINT timer interval. All statistics values are set to NULL, except for NACTIVE, SPACE and EXTENTS. The statistics values will be set after the first REORG, RUNSTATS or COPY.

The externalize process is made by the RTS manager. It runs under a system task in DBM1 address space that is created during DB2 start up. The CPU used by the RTS manager is included in DBM1’s SRB time. RTS manager is triggered on a time interval defined by the system parameter STATSINT. It does the following tasks:

- Orders the active statistics blocks in clustering order
- Inserts/updates the rows in the RTS tables through the clustering index
- Frees the dormant statistics blocks that belong to data sets being closed

6.4.2  When real time statistics are updated

In general INSERT, UPDATE, and DELETE statements cause DB2 to modify the real time statistics. However, certain DB2 utilities also affect the statistics (Figure 6-1).
Table 6-3 shows how SQL operations affect the counter columns in real time statistics. Counter columns are the columns that record the number of insert, delete, or update operations. The total counters TOTALROWS and TOTALENTRIES record the number of rows in the page set.

When you perform a mass delete operation on a table space, DB2 does not reset the counter columns in the real time statistics tables.

Table 6-3  Updating real time statistics

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>Rolled-back statement</th>
<th>Incremented counters</th>
</tr>
</thead>
<tbody>
<tr>
<td>UPDATE</td>
<td>-</td>
<td>Update counters</td>
</tr>
<tr>
<td>INSERT</td>
<td>-</td>
<td>Insert counters</td>
</tr>
<tr>
<td>DELETE</td>
<td>-</td>
<td>Delete counters</td>
</tr>
<tr>
<td>ROLLBACK</td>
<td>UPDATE</td>
<td>Update counters</td>
</tr>
<tr>
<td>INSERT</td>
<td></td>
<td>Delete counters</td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
<td>Insert counters</td>
</tr>
</tbody>
</table>

6.4.3 Real time statistics and DB2 utilities

Table 6-4 summarizes how utilities affect real time statistics in a general way. For complete details on the columns affected and the scenarios, see Appendix G of DB2 UDB for OS/390 and z/OS Version 7 Administration Guide, SC26-9931-01.
### Table 6-4  Real time statistics and DB2 utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Effect in RTS</th>
</tr>
</thead>
</table>
| REORG                         | ▶  Set REORGLASTTIME  
  ▶  Reset REORG related statistics  
  ▶  Log apply changes for online REORG will be treated as Inserts/Deletes/Updates |
| RUNSTATS                      | ▶  Set STATSLASTTIME  
  ▶  Reset RUNSTATS related statistics                                       |
| COPY                          | ▶  Set COPYLASTTIME  
  ▶  Reset COPY related statistics                                              |
| LOAD REPLACE                  | ▶  Set LOADRLASTTIME  
  ▶  Reset REORG related statistics                                              |
| LOAD REPLACE PART or REORG PART | ▶  Do not reset REORG statistics for non-partitioned indexes (NPI)  
  ▶  Statistics for NPIs are updated as INSERT and DELETE                      |
| COPY with DSNUM option         | ▶  Do not reset COPYLASTTIME  
  ▶  Do not reset COPY related statistics                                         |
| RECOVER TORBA/TOCOPY          | ▶  Set REORGLASTTIME, STATSLASTTIME, COPYLASTTIME, LOADRLASTTIME and REBUILDLASTTIME to NULL  
  ▶  Reset REORG, RUNSTATS and COPY statistics to NULL                           |
| REBUILD INDEX                 | ▶  Set REBUILDLASTTIME  
  ▶  Reset REORG related statistics                                              |
| Online LOAD RESUME            | ▶  Treated as INSERTS                                                       |

### 6.4.4 Real time statistics and non-DB2 utilities

Non-DB2, or offline utilities currently do not affect real time statistics. However, an object that is the target of a non-DB2 COPY, LOAD, REBUILD, REORG or RUNSTATS can cause incorrect statistics to be inserted in the real time statistics tables. Follow this process, unless otherwise informed, to ensure correct statistics when you run non-DB2 utilities:

1. Stop the table space or index on which you plan to run the utility to externalize the in-memory statistics.
2. Run the utility.
3. When the utility completes, update the statistics tables with new totals, timestamps, and zero incremental counter values. Subsequent SQL operations updating in-memory statistics generate correct values with respect to the utility end point.

### 6.4.5 Real time statistics in data sharing

Each member of the data sharing group updates its statistics serially. Each one reads the target row from statistics table, obtains a lock, aggregates its in-memory statistics, and updates the statistics table with the new totals. Each member sets its own interval for writing real time statistics.
DB2 performs locking based upon the lock size of the DSNRTSDB.DSNRTSTS table space. So, this is the reason to use LOCKSIZE ROW. Reading the statistics tables uses ISOLATION CS and CURRENTDATA YES semantics.

Utilities that reset page sets to empty can invalidate other DB2 member’s in-memory statistics. The member that resets a page set will notify the other DB2 members that a reset has occurred and the in-memory statistics are invalidated. If the notify process fails, the utility running on the “resetting” member will not fail. The appropriate timestamp (REORGLASTTIME, STATSLASTTIME or COPYLASTTIME) is set to NULL to indicate the table statistics are unknown.

### 6.4.6 Accuracy of the statistics

In general the real time statistics are accurate values. But several factors can cause the tables values to become inaccurate:

- Certain utility restart scenarios
- DB2 subsystem failure
- DB2 stopped with STOP DB2 MODE(FORCE)
- Notify failure in a data sharing environment
- Running third party utilities without flushing the in-memory statistics

To restore accurate values, you need to run REORG, RUNSTATS and COPY on the suspect object.

### 6.5 DSNACCOR stored procedure

DSNACCOR is a sample stored procedure that returns a list of table spaces as result set, using the data from RTS tables, IFI calls, and parameters passed to the stored procedure. It helps to determine which DB2 objects:

- Should be reorganized
- Should have their statistics updated
- Should be image copied
- Have exceeded number of extents
- Are in a restricted status

DSNACCOR by default uses generic rules of thumb (ROT) thresholds and limits that may not be suitable for your installation. You can customize any threshold or limit (33 input options) to fit your system or environment. Also, the DSNACCOR’s focus is system wide, but you can change the scope of its queries by using a WHERE clause and/or an exception table. This additional filter provided by the WHERE clause will be used when creating the return cursor.

DSNACCOR must run in a WLM-established stored procedure address space and it creates and uses declared temporary tables. So the target DB2 must have the TEMP database defined. Select authority on the RTS tables and DISPLAY system privilege are also required for the owner of the DSNACCOR package.

For a detailed information of DSNACCOR, like description of the input parameters, output parameters, result set and formulas used for recommending actions see Appendix H of DB2 UDB for OS/390 and z/OS Version 7 Administration Guide, SC26-9931-01.
6.6 Performance considerations

- To have better performance, we recommend that you put the RTS objects in an isolated buffer pool. This will increase the performance of the RTS manager accessing and updating these objects.

- Avoid time-outs and deadlocks with RTS manager when accessing RTS tables using uncommitted read (UR) lock isolation whenever possible. Avoid using SHRLEVEL CHANGE when running REORG, RUNSTATS and COPY on the RTS objects.

- The impact on CPU of the code increase in DB2 due to incrementing and decrementing row, LOB, or index counts has been measured as minimal (that is less than 5%). Also, the code increment to initialize the RTS control block during the first update is negligible. Inserting and updating the table statistics has a negligible impact, since the task is asynchronous to DB2.
In this chapter, we discuss the following topics:

- V1.1 to V7.1 performance comparisons: The new DB2 OLAP Server for OS/390 V7.1 offers significant functional and performance enhancements over the previous release.

- Throughput and scalability measurements: The characteristics of zSeries allow multiple cubes to be loaded and calculated concurrently, enabling greater throughput within batch windows.
7.1 DB2 OLAP Server introduction

DB2 OLAP Server is an Online Analytical Processing (OLAP) product that can be used for a wide range of multidimensional reporting, analysis, modeling, and planning applications. OLAP applications are designed for business analysts who need rapid response to complex data analysis with multidimensional calculations. They are typically used in the financial and marketing area.

The main concept of the OLAP architecture is that the data is pre-summarized, pre-calculated and stored in a structure often referred to as a cube. First the data is loaded into the cube. DB2 OLAP server can load data from various types of flat files or directly from relational database systems. Then the calculation process applies the formulas, aggregations, and analytic functions that were defined when the cube was designed. The additional information is stored into the cube, enabling subsequent queries to run very quickly.

The process of loading and calculating a cube is quite similar to traditional batch processing: it can be a long-running process that is generally executed during non-production hours. Calculations can be especially long running and can impose heavy CPU, I/O, and memory requirements. Some cubes require hours to calculate and must run during a batch window.

DB2 OLAP Server for OS/390 offers two options to store the cubes on the server. With DB2 OLAP Server for OS/390 V7.1, you can choose the storage option (RSM or MSM) based on the individual application’s needs:

- **Using Relational Storage Manager (RSM):** OLAP stores the data in DB2 relational tables. You should consider this option if your primary interest is flexibility. Since the data is stored in relational tables, users can use SQL to access the OLAP cubes and system administrators can take advantage of DB2 utilities for backup and recovery.

- **Using Multidimensional Storage Manager (MSM):** OLAP stores the cubes in Essbase’s proprietary file structure, which is highly optimized for best performance. With DB2 OLAP Server for OS/390 V7.1, MSM uses VSAM linear data sets (versus HFS files in V1.1). You should consider this option if your batch window is constrained and performance for load and calculation processes is a critical requirement for your application.

A set of performance measurements were executed at the Silicon Valley Laboratory with two objectives:

- Evaluating the performance benefits of the new DB2 OLAP Server for OS/390 V7.1 over the previous release V1.1
- Measuring the throughput and scalability characteristics of DB2 OLAP Server V7.1 in a large zSeries environment

All the performance measurements were done using MSM (that is without using DB2.) We summarize the results of the measurements in this chapter because they show that by using the MSM option based on VSAM, DB2 OLAP Server for OS/390 V7.1 provides a high-performance solution simple to implement. This solution also takes advantage of running DB2 OLAP in a zSeries environment by:

- Minimizing the cost and effort of moving data to other servers for analysis
- Ensuring high availability of mission-critical OLAP applications
- Using Workload Manager (WLM) to dynamically and effectively balance system resources
- Using existing S/390 platform skills.

For a more detailed evaluation of DB2 OLAP Server performance on zSeries, you can ask your IBM representative to consult the white paper *DB2 OLAP Server for OS/390 Performance* by Ron Yorita, currently available at the internal Web site:

http://w3.ibm.com/software/data
7.2 V1.1 to V7.1 performance comparisons

A set of measurements were executed to compare DB2 OLAP Server V7.1 to the previous release V1.1:

- The first comparisons were conducted on identical hardware. These measurements demonstrate the performance improvements of DB2 OLAP Server V7.7 over V1.1, and show the benefit customers may see by migrating to DB2 OLAP Server for OS/390 V7.1.

- Subsequent comparisons were performed with DB2 OLAP Server for OS/390 V1.1 running on hardware that customers may have been using when the product was initially available versus DB2 OLAP Server for OS/390 V7.7 running on current hardware. These measurements demonstrate the benefit customers may see by upgrading the current hardware as well as migrating to DB2 OLAP Server for OS/390 V7.1.

7.2.1 Measurement environment

The performance measurements were conducted using the following hardware and software:

- Host environment I:
  - S/390 9672 G6 processor
  - 1 Logical Partition (LPAR) with 4 dedicated CPUs
  - 1.8 GB real memory, 6 GB expanded

- Host environment II:
  - zSeries 900 processor
  - 1 LPAR with 2 dedicated CPUs
  - 2 GB real memory, 6 GB expanded

- Disks:
  - RAMAC Virtual Array (RVA)
  - ESS Model E20 with ESCON channels
  - ESS Model F20 with FICON channels

- Software:
  - OS/390 V2R10
  - DB2 OLAP Server for OS/390 V1.1
  - DB2 OLAP Server for OS/390 V7.1 PTF1

For performance monitoring and analysis purposes, all measurements were run single-threaded, in an environment where there was no other activity. The cube index and data were isolated on separate devices. The load was from a single flat file (no rule file and no incremental load) located in an Unix System Services (USS) file system, again isolated on a separate device.

7.2.2 Measurement results

The first series of measurements were performed on a G6 LPAR using different I/O subsystem hardware:

- RVA
- ESS Model E20 with ESCON channels
- ESS Model F20 with FICON channels

Six different cubes were used for the measurements: three cubes were derived from customer cubes, three were internal to IBM. Figure 7-1 illustrates the accumulated load and calculation times for the six cubes in each environment.
In all the cases, DB2 OLAP Server for OS/390 V7.1 represents a significant progression compared to DB2 OLAP Server for OS/390 V1.1:

- 37% improvement using RVA
- 44% improvement using ESS Model E20 with ESCON channels
- 50% improvement using ESS Model F20 with FICON channels

Moreover, DB2 OLAP Server V7.1, because it uses VSAM linear data sets instead of HFS, takes better advantage of the recent enhancements of I/O subsystem technology.

Note that all the measurements with DB2 OLAP Server V1.1 were done with a 4 KB index page size. It is not the default, but it was found that the default index page size of 1 KB was not optimal for V1.1. and that V1.1 performs better with a 4 KB index page size. Therefore, if you are migrating from V1.1 (using the default 1 KB index page size) to V7.1, you may experience even greater differences. Measurements were performed to estimate the gain. A 60% improvement in accumulated load and calculation times was measured using RVA disks and five cubes; a 32% improvement was measured when using V1.1 with a 4 KB index page size instead of V1.1 with a 1 KB index page size.

Note: With DB2 OLAP Server V7.1, the index page size does not impact performance.

Subsequent comparisons were performed with DB2 OLAP Server for OS/390 V1.1 and OLAP Server for OS/390 V7.1 running on different types of hardware. These measurements were run with the specific objective to demonstrate the benefit customers may see by upgrading the current hardware as well as migrating to DB2 OLAP Server for OS/390 V7.1.

The following environments were compared:

1. V1.1 on G6 processor and RVA, using the non-optimal 1 KB index page size
2. V1.1 on G6 processor and RVA
3. V1.1 on G6 processor and ESS Model E20 with ESCON channels
4. V1.1 on G6 processor and ESS Model F20 with FICON channels
5. V7.1 on G6 processor and RVA
6. V7.1 on G6 processor and ESS Model E20 with ESCON channels
7. V7.1 on G6 processor and ESS Model F20 with FICON channels
8. V7.1 on zSeries 900 processor and ESS Model F20 with FICON channels

Figure 7-2 shows the load and calculation times for four cubes in each environment.

![DB2 OLAP Server for OS/390 V1.1 versus V7.1](chart)

Figure 7-2  DB2 OLAP Server for OS/390 V1.1 versus V7.1 using different type of hardware

See the first bar for V1.1 and the last bar for V7.1. This shows what V1.1 customers may see when they upgrade to the latest technology (zSeries 900 processor and ESS Model F20 with FICON channels) and migrate to DB2 OLAP Server V7.1. The range of improvement is from 2.5 times to 6.7 times for these cubes.

**Note:** For all the measurements, the load of the cube was done using the Remote/Server data file location option, in order to bypass the TCP/IP penalty.

When you load a cube from a flat file, you can choose one of the following options:

1. Local/client
2. Remote/server data file
3. File

The File option may be the most common method of loading as it allows you to specify the full path name of the input file. However, if the file is located on the server, it causes the server to pass the data to OLAP using TCP/IP loopback.

Therefore, for optimal load performance, using the Remote/Server data file option is recommended when using an input file located on the server. The Remote/Server File option requires the input file to be on the same system as the DB2 OLAP Server and in the $ARBORPATH/app/<app>/<db> directory (a symbolic link to the file is sufficient).
7.3 Throughput and scalability measurements

Some customer installations have hundreds of cubes that require periodic loads and calculations. The characteristics of zSeries allow multiple cubes to be loaded and calculated concurrently, enabling greater throughput within batch windows. This section describes measurements that demonstrate the throughput and scalability characteristics of DB2 OLAP Server V7.1 on the zSeries platform.

7.3.1 Measurement environment

The performance measurements were conducted using the following hardware and software:

- **Host environment:**
  - zSeries 900 processor
  - 1 LPAR with 12 dedicated CPUs
  - 48 GB real memory – no expanded memory

- **DASD:**
  - ESS Model F20 with 16 ESCON channels
  - In one case, a second ESS Model F20 with 16 ESCON channels

- **Software:**
  - OS/390 V2R10 – z/Architecture (64-bit mode)
  - DB2 OLAP Server for OS/390 V7.1 PTF3

Three cubes were used. Table 7-1 summarizes for each cube the elapsed times for single-threaded load and calculation, as well as the size of the final calculated cube (data only, the index size is not included). This is the base for subsequent concurrent load and calculation comparisons.

<table>
<thead>
<tr>
<th>Cube 1</th>
<th>Load Elapsed time</th>
<th>Calculation Elapsed time</th>
<th>Final cube size (data only)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>61 sec</td>
<td>394 sec</td>
<td>868 MB</td>
</tr>
<tr>
<td>Cube 2</td>
<td>88 sec</td>
<td>1,246 sec</td>
<td>1.212 MB</td>
</tr>
<tr>
<td>Cube 3</td>
<td>50 sec</td>
<td>247 sec</td>
<td>311 MB</td>
</tr>
</tbody>
</table>

To run multiple concurrent loads and calculations, copies of a given cube were created as separate OLAP applications and databases. For performance monitoring and analysis purposes, all scenarios were homogeneous using multiple copies of a given cube. The cubes were spread across the LCUs. The two VSAM data sets corresponding to the cube data and index were isolated on separate devices within a LCU. All loads were from single flat files (no rule files and no incremental loads), again isolated on separate devices. The Remote/Server File option was used to optimize the loads.

7.3.2 Measurement results

In this section, we provide the results of the scalability measurements.

Concurrent calculations

Figure 7-3 shows the average calculation time for each of the three cubes, with varying number of concurrent calculations.
In the case of cube1, there are two curves:
- The first test was conducted using a single ESS with a total of 16 ESCON channels
- The second test was performed using two ESSs with a total of 32 ESCON channels

![Average calculation time of the cubes](image1)

Figure 7-3 Average calculation time of the cubes

Figure 7-4 shows the CPU utilization of the system, with a varying number of concurrent calculations.

![CPU utilization during calculation](image2)

Figure 7-4 CPU utilization during the calculation of the cubes
Cube1 is I/O bound. Figure 7-3 on page 103 shows that the average calculation time is fairly constant with up to 12 cubes. At 16 cubes, the environment is becoming I/O constrained. By adding a second ESS and spreading the cubes across both ESSs, the constraint is relieved and the curve remains much flatter. RMF data indicates that the degradation is due to the ESCON channels which become saturated.

With only one ESS, the average calculation time increases by 30% with 20 cubes (512 versus 394 sec for one cube), which is 1.5% per cube. With two ESSs and 20 cubes, there is only a 19% increase in the calculation time (466 versus 394 sec), which is less than 1% per cube. Figure 7-4 shows that the CPU utilization is 80-83% with 20 cubes. The curves are both relatively linear. The divergence after 12 cubes shows the effect of the I/O constraint.

Cube2 is CPU bound. Figure 7-3 shows that the average calculation time is almost constant with up to 12 cubes, indicating that there are no constraints. The calculation time is 8% higher with 12 cubes than with one (1,344 versus 1,246 sec), which is less than 1% per cube. Figure 7-4 shows that with 12 cubes, the CPU utilization of the 12-CPU LPAR is 88%. With the remaining CPU capacity, it would be possible to run a 13th cube2. However, 14 concurrent calculations would be CPU constrained.

Cube3 is less I/O-bound than cube1 and less CPU-bound than cube2. Figure 7-3 shows that the average calculation time is fairly flat with up to 12 cubes. At 16 cubes, the environment is becoming slightly I/O constrained. Using a second ESS along with the additional channels would have eliminated the problem. However, even with only a single ESS, the degradation is less than 2% per cube at 16 cubes (308 versus 247 sec). Figure 7-4 shows that the CPU usage increases linearly up to 12 cubes. At 16 cubes, there is a slight drop because of the emerging I/O constraint.

We use the number of cubes calculated per hour as a throughput metric to evaluate the benefit of calculating cubes concurrently. In the case of this study, the following improvements were measured:

- 15.4 times increase when running 20 cube1 calculations versus one with one ESS.
- 16.9 times increase when running 20 cube1 calculations versus one with two ESSs.
- 11.1 times increase when running 12 cube2 calculations versus one.
- 12.8 times increase when running 16 cube3 calculations versus one.

**Concurrent loads**

Figure 7-5 shows the average load time for each of the three cubes, with a varying number of concurrent loads. The maximum number of concurrent loads was determined by the maximum number of calculations.
Figure 7-5  Average load time of the cubes

Figure 7-6 shows the CPU utilization of the system, with varying number of concurrent loads.

Figure 7-6  CPU utilization during the load of the cubes

Beyond 12 concurrent cubes, the average load time begins to increase noticeably for cube1 and cube3 because of CPU constraints (see Figure 7-6). Isolating the input files and cubes onto separate devices and LCUs has made the load process CPU-bound.

When there are no bottlenecks (up to 12 concurrent loads), the loads of the three cubes have a very comparable behavior. With 12 cubes, the average elapsed time increases by 5 to 8% compared with a single load, which is less than 1% per cube. The throughput of 12 concurrent loads is 11 times higher than this of a single load.
64-bit versus 31-bit architecture

DB2 OLAP Server can be memory-intensive and real memory can rapidly be over-committed if multiple cubes are concurrently calculated. Each cube has its own set of caches and over-sizing the cache sizes has a multiplicative effect. For example, some single cube calculation measurements were run where the total cache size (index, data, and file caches) was 670 MB per cube. If 20 concurrent calculations were executed with the same cache settings, the DB2 OLAP Server caches alone would require $670 \text{ MB} \times 20 = 13,400 \text{ MB}$ of real memory. Aside from DB2 OLAP Server caches, code, buffers, control structures, other users, applications, subsystems, and the operating system require real storage.

For the throughput and scalability tests, the cache settings were reduced from 670 MB down to 190 MB per cube. Cutting the total cache setting by more than 3.5 times resulted in a minimal increase in the elapsed time (less than 5%). For the 20 concurrent cube calculations, using these reduced settings the DB2 OLAP Server caches require $190 \text{ MB} \times 20 = 3,800 \text{ MB}$. Thus, with minimal impact to calculation times, the cache storage requirement was reduced from 13.4 GB down to 3.8 GB.

All the preceding throughput measurements were run in an 64-bit architecture environment configured with 48 GB of real storage. In this environment, there were no memory constraints. However, in a 31-bit architecture, where the maximum amount of real storage that can be configured is 2 GB, the memory would be over-committed.

Figure 7-7 shows the average calculation times running in 31-bit and 64-bit architectures. The divergence of the two curves after 8 concurrent calculations shows the effects of over-committing memory.

![Figure 7-7   Average calculation time: 64-bit versus 31-bit](image)

At 12 cubes, the expanded storage paging rate is 1,100 pages/sec. As a result, the average calculation time is 17% higher than the one measured in the 64-bit architecture. At 20 cubes, the expanded storage paging rate is 19,250 pages/sec and the average calculation time has degraded by 64% compared with 64-bit mode.
To achieve optimal performance when running concurrent calculations, the cache settings for each cube should be reduced to minimize storage requirements until calculation times begin to increase. Avoid over-committing real storage by controlling the number of concurrent calculations. Run with 64-bit architecture and configure the system with ample real storage.

### 7.4 Summary

The measurements reported in this chapter demonstrate the improvements that DB2 OLAP Server V7.1 provides over the previous release. The new VSAM implementation for MSM enables DB2 OLAP Server to exploit the latest I/O subsystem technology far more effectively than V1.1, allowing larger cubes to be built within the same batch window.

The measurements also demonstrate that in a well-tuned environment, multiple concurrent loads and calculations can be run with minimal increases in elapsed times. The number of concurrent loads and calculations depends upon the cube characteristics and the resources available (number of CPUs, I/O subsystem, and memory).

The following components must be carefully planned and controlled:

- VSAM data sets and HFS placement is critical to avoid I/O bottlenecks.
- Cache settings must be minimized to avoid over-committing real memory.

Using the 64-bit architecture along with the current I/O subsystem hardware with sufficient channels is strongly recommended.
Recent performance and availability maintenance

In this appendix, we look at DB2 V5, V6, and V7, as well as DB2 PM, SDK, and z/OS recent maintenance that generally relates to DB2 performance and availability.

This list represents a snapshot of the current maintenance at the moment of writing, and as such, it becomes incomplete or even incorrect at the time of reading. It is here to identify areas of performance improvements. Make sure to check on RETAIN the applicability of these APARs to your environment, and to verify pre- and post-requisites.
A.1 DB2 for OS/390 V5 APARs

In Table A-1 we briefly describe the DB2 performance related APARS for DB2 V5.

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ48905</td>
<td>Complex views</td>
<td>This corrects prepare execution problems with complex views with table expressions in outer joins. It applies to DB2 V5 and V6 as well.</td>
</tr>
<tr>
<td>PQ49173</td>
<td>Data sharing</td>
<td>Premature counting of buffer pool Data Management threshold (DMTH) reached in data sharing.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>Load and reorg utilities</td>
<td>This reduces the elapsed time for the catalog updates when executing LOAD LOG NO and REORG LOG NO jobs on many partitions. PTF available for DB2 V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ51845</td>
<td>SQLJ/JDBC</td>
<td>This fixes several reported defects that are applicable to all DB2 for OS/390 SQLJ/JDBC driver users. PTFs exist for V6 and V7.</td>
</tr>
<tr>
<td>PQ54754</td>
<td>SQLJ/JDBC</td>
<td>This fixes errors and helps performance.</td>
</tr>
<tr>
<td>PQ44580</td>
<td>CPU parallelism</td>
<td>This eliminates loop in queries utilizing CPU parallelism. For V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ47767</td>
<td>COBOL</td>
<td>This ensures that related host variables are recognized properly. Valid for V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ47582</td>
<td>Utilities</td>
<td>Correction to restart functions of RELOAD. It applies to DB2 V6 and V7 as well.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>Utilities</td>
<td>Needed after the PTF for PQ47582 to reduce increased elapsed time when dealing with many partitions. It applies to V6 and V7 as well.</td>
</tr>
</tbody>
</table>

A.2 DB2 for OS/390 V6 APARs

In Table A-2 we briefly describe the DB2 performance related APARS for DB2 V6.

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>II12836</td>
<td>Query parallelism</td>
<td>DB2 V6 and V7 Query parallelism recommended maintenance</td>
</tr>
<tr>
<td>PQ47914</td>
<td>CPU parallelism</td>
<td>This limits the maximum degree of parallelism in a DB2 environment therefore reducing the consumption of virtual storage</td>
</tr>
<tr>
<td>PQ44580</td>
<td>CPU parallelism</td>
<td>This eliminates loop in queries utilizing CPU parallelism. For V5, V6, and V7</td>
</tr>
<tr>
<td>PQ45820</td>
<td>CPU parallelism</td>
<td>This eliminates performance overhead for short running parallel queries when compared to sequential execution. It sets differently the parallelism threshold ZPARM (SPRMPTH) introduced in PQ25135.</td>
</tr>
<tr>
<td>PQ44103</td>
<td>Instrumentation Facility</td>
<td>This fixes incorrect numbers in IFCID 0003, IFCID 0018, IFCID 0058 and IPCS DSNWDMP formatting routine for data sharing.</td>
</tr>
<tr>
<td>APAR</td>
<td>Area</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PQ47970</td>
<td>Instrumentation Facility</td>
<td>This corrects accumulated TCB and elapsed time in accounting records while executing nested triggers.</td>
</tr>
<tr>
<td>PQ47973</td>
<td>Instrumentation Facility</td>
<td>This adds a new IFCID 0234 to return user authorization information.</td>
</tr>
<tr>
<td>PQ48031</td>
<td>Data space</td>
<td>Lookaside buffer pool limit is increased from 16 MB to 256 MB. APAR PQ53070 is also needed.</td>
</tr>
<tr>
<td>PQ48126</td>
<td>Log size on disk</td>
<td>Increase maximum active and archive log data set size from 2 to 4 GB.</td>
</tr>
<tr>
<td>PQ49328</td>
<td>Data sharing</td>
<td>Improved tree P-lock interDB2 negotiation. For V6 and V7.</td>
</tr>
<tr>
<td>PQ54451</td>
<td>Data sharing</td>
<td>Deadlock reduction with command issuing. V7 as well.</td>
</tr>
<tr>
<td>PQ55682</td>
<td>Data sharing</td>
<td>Locking is reduced across group members when catalog migration and catalog reorg processing is taking place. It applies to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ48522</td>
<td>Outer join</td>
<td>This reduces the instances of materialization for outer join when scanning table expression view or table expression merge.</td>
</tr>
<tr>
<td>PQ52142</td>
<td>Outer join</td>
<td>This makes sure that frequency statistics are always used for materialized outer joins</td>
</tr>
<tr>
<td>PQ51765</td>
<td>Star join</td>
<td>This provides users with a new parameter SJTABLES in DSNTIJUZ to control the star join enablement threshold introduced by APARs PQ43846 (V6) and PQ47833 (V7).</td>
</tr>
<tr>
<td>PQ37880</td>
<td>Global temporary tables</td>
<td>This correction reduces storage allocation in the DBD with thread reusing.</td>
</tr>
<tr>
<td>PQ43846</td>
<td>Star Join</td>
<td>This fixes performance degradation for some star schema queries when star schema join method (star join) is enabled</td>
</tr>
<tr>
<td>PQ48905</td>
<td>Complex views</td>
<td>This corrects prepare execution problems with complex views with table expressions in outer joins. It applies to DB2 V5 and V7 as well.</td>
</tr>
<tr>
<td>PQ49121</td>
<td>Complex joins</td>
<td>This allows PREPARE and BIND on a number of tables being joined larger than 15. It works in conjunction with a ZPARM change. See also PQ31326. It also applies to DB2 V7.</td>
</tr>
<tr>
<td>PQ55663</td>
<td>Complex joins</td>
<td>This reduces performance degradation of PREPARE for a 27 tables join. It applies to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ51846</td>
<td>SQL/JDBC</td>
<td>It fixes several reported defects that are applicable to all DB2 for OS/390 SQL/J/JDBC driver users. PTFs exist for V5 and V7.</td>
</tr>
<tr>
<td>PQ54580</td>
<td>Access path</td>
<td>This adds checks to find the best matching column candidates.</td>
</tr>
<tr>
<td>PQ54755</td>
<td>SQL/JDBC</td>
<td>This fixes errors and helps performance</td>
</tr>
<tr>
<td>PQ47767</td>
<td>COBOL</td>
<td>This ensures that related host variables are recognized properly. Valid for V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ51078</td>
<td>COBOL</td>
<td>This reduces CPU consumption by dealing differently with host variables. PTF available for V7 as well. It requires PQ53359, PQ56704, and PQ57331 to improve CPU time.</td>
</tr>
</tbody>
</table>
In Table A-3 we briefly describe the DB2 performance related APARS for DB2 V7.

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ55247</td>
<td>SQL Procedures</td>
<td>This translates some SQL Procedure statements to C language rather than SQL. It applies to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ55446</td>
<td>SQL Procedures</td>
<td>After PQ55247, it translates more SQL Procedure statements to C language rather than SQL. It applies to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ60025</td>
<td>SQL Procedures</td>
<td>Needed after PQ55247 and PQ55446 to resolve some 0C4. For DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ58820</td>
<td>LOBs</td>
<td>This provides availability, performance, and integrity improvements for LOBs.</td>
</tr>
<tr>
<td>PQ54552</td>
<td>RLF</td>
<td>This resolves premature -905 when UDF is used. It applies to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ53565</td>
<td>Workfiles</td>
<td>This reduces occurrences of unavailable resource. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ53571</td>
<td>VSAM striped data sets</td>
<td>This allows striping for table spaces defined with 4 KB pages and it provides better performance when preformatting. V7 as well.</td>
</tr>
<tr>
<td>PQ50106</td>
<td>Load utility</td>
<td>LOAD with INLINE STATISTICS has improved performance by avoiding unnecessary catalog accesses for getting the column names.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>Load and reorg utilities</td>
<td>This reduces the elapsed time for the catalog updates when executing LOAD LOG NO and REORG LOG NO jobs on many partitions. PTF available for DB2 V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ43509</td>
<td>Utilities</td>
<td>High not accounted time in class 3 accounting for utilities running with data space buffer pools.</td>
</tr>
<tr>
<td>PQ45176</td>
<td>BUILD2 phase</td>
<td>The parallel BUILD2 phase of REORG SHRLEVEL CHANGE or REFERENCE eliminates logging of non partitioning index logical part. Applicable to DB2 V7 as well.</td>
</tr>
<tr>
<td>PQ47582</td>
<td>Utilities</td>
<td>Correction to restart functions of RELOAD. It applies to DB2 V5 and V7 as well.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>Utilities</td>
<td>Needed after the PTF for PQ47582 to reduce increased elapsed time when dealing with many partitions. It applies to V5 and V7 as well.</td>
</tr>
<tr>
<td>PQ50332</td>
<td>DSN1COPY</td>
<td>Performance is improved by using SPEED in AMD DEFINE CLUSTER for DB2 managed data sets. It applies also to DB2 V7.</td>
</tr>
<tr>
<td>PQ55078</td>
<td>storage</td>
<td>This helps reduce application storage below the line with large number of tasks. It applies to V7 as well.</td>
</tr>
<tr>
<td>PQ55346</td>
<td>storage</td>
<td>This improves the storage shortage cushion. It applies to V7 as well.</td>
</tr>
</tbody>
</table>

**A.3 DB2 for OS/390 V7 APARs**

In Table A-3 we briefly describe the DB2 performance related APARS for DB2 V7.
<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>II12836</td>
<td>Query parallelism</td>
<td>DB2 V6 and V7 Query parallelism recommended maintenance</td>
</tr>
<tr>
<td>PQ44580</td>
<td>CPU parallelism</td>
<td>This eliminates loop in queries utilizing CPU parallelism. For V5, V6, and V7</td>
</tr>
<tr>
<td>PQ44671</td>
<td>Data sharing</td>
<td>Support for CURRENT MEMBER special register has been added. This helps with application requirements and performance by avoiding interDB2 read/write interest. CURRENT MEMBER specifies the member name of the current DB2 data sharing member on which the statement is executing. The data type is CHAR(8) padded on the right with blanks if necessary.</td>
</tr>
<tr>
<td>PQ49328</td>
<td>Data sharing</td>
<td>Improved tree P-lock interDB2 negotiation.</td>
</tr>
<tr>
<td>PQ54451</td>
<td>Data sharing</td>
<td>Deadlock reduction with command issuing. V6 as well.</td>
</tr>
<tr>
<td>PQ55682</td>
<td>Data sharing</td>
<td>Locking is reduced across group members when catalog migration and catalog reorg processing is taking place. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ63430</td>
<td>Data sharing</td>
<td>After PQ60038, a looping on DELETE_NAME calls to the CF is eliminated.</td>
</tr>
<tr>
<td>PQ45526</td>
<td>Scrollable cursors</td>
<td>This reduces locking with scrollable cursors when using ISOLATION CS and CURRENTDATA(NO).</td>
</tr>
<tr>
<td>PQ47178</td>
<td>UNION</td>
<td>This enhancement reduces the number of workfiles needed for materialization for UNION ALL type of queries by trying to avoid materialization of UNION ALL results in view or table expression.</td>
</tr>
<tr>
<td>PQ55393</td>
<td>UNION</td>
<td>This enhancement improves performance of UNION in view with host variables.</td>
</tr>
<tr>
<td>PQ63825</td>
<td>Global temporary tables</td>
<td>This correction reduces storage allocation in the ADMF pool.</td>
</tr>
<tr>
<td>PQ49121</td>
<td>Complex joins</td>
<td>This allows PREPARE and BIND on a number of tables being joined larger than 15. It works in conjunction with a ZP ARM change. See also PQ31326. It also applies to DB2 V6.</td>
</tr>
<tr>
<td>PQ56332</td>
<td>Global temporary tables</td>
<td>This correction reduces storage allocation in the EDM pool due to better DBD storage management.</td>
</tr>
<tr>
<td>PQ47833</td>
<td>Star join</td>
<td>This fixes performance degradation for some star schema queries when star schema join method (star join) is enabled</td>
</tr>
<tr>
<td>PQ51765</td>
<td>Star join</td>
<td>This provides users with a new parameter SJTABLES in DSNTIJUZ to control the star join enablement threshold introduced by APARs PQ43846 (V6) and PQ47833 (V7).</td>
</tr>
<tr>
<td>PQ61458</td>
<td>Star join</td>
<td>This improves performance for star join with snowflakes by eliminating merge join and using sparse indexes.</td>
</tr>
<tr>
<td>APAR</td>
<td>Area</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PQ48306</td>
<td>Outer join</td>
<td>DB2 V7 contains changes in the access path selection algorithm for queries with inner joins of a large number of tables. Code was changed to include queries with outer and mixed inner and outer joins in the improved access path selection algorithms and reduced storage requirements.</td>
</tr>
<tr>
<td>PQ48905</td>
<td>Complex views</td>
<td>This corrects prepare execution problems with complex views with table expressions in outer joins. It applies to DB2 V5 and V6 as well.</td>
</tr>
<tr>
<td>PQ55663</td>
<td>Complex joins</td>
<td>This reduces performance degradation of PREPARE for a 27 tables join. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ58420</td>
<td>Access path</td>
<td>This introduces matching index access for IS NOT NULL predicate.</td>
</tr>
<tr>
<td>PQ58843</td>
<td>Locking</td>
<td>This eliminates occurrence of deadlocks for UPDATE CURSOR with CS ISOLATION.</td>
</tr>
<tr>
<td>PQ51847</td>
<td>SQL/JDBC</td>
<td>This fixes several reported defects that are applicable to all DB2 for OS/390 SQL/JDBC driver users. PTFs exist for V5 and V6.</td>
</tr>
<tr>
<td>PQ54756</td>
<td>SQL/JDBC</td>
<td>This fixes errors and helps performance</td>
</tr>
<tr>
<td>PQ54042</td>
<td>BETWEEN predicate</td>
<td>Performance improvement related to the BETWEEN predicate by additional check to find best matching column candidates.</td>
</tr>
<tr>
<td>PQ54552</td>
<td>RLF</td>
<td>This resolves premature -905 when UDF is used. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ53565</td>
<td>Workfiles</td>
<td>This reduces occurrences of unavailable resource. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ55247</td>
<td>SQL Procedures</td>
<td>This translates some SQL Procedure statements to C language rather than SQL. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ55446</td>
<td>SQL Procedures</td>
<td>After PQ55247, it translates more SQL Procedure statements to C language rather than SQL. It applies to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ60025</td>
<td>SQL Procedures</td>
<td>Needed after PQ55247 and PQ55446 to resolve some 0C4. For DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ60839</td>
<td>RENAME TABLE</td>
<td>This eliminates performance degradation when renaming a table.</td>
</tr>
<tr>
<td>PQ53571</td>
<td>VSAM striped data sets</td>
<td>This allows striping for table spaces defined with 4 KB pages and it provides better performance when preformatting. V6 as well.</td>
</tr>
<tr>
<td>PQ54580</td>
<td>Data set management</td>
<td>This is an enhancement to the VSAM REUSE function. Support is added for striped VSAM data sets. Also, for VSAM extended format data sets, the REUSE function is enhanced to partially release space from the primary volume, and to support the resetting of indexes for VSAM Key Sequenced Data Sets (KSDS). APAR OW50528 is a prerequisite for this function.</td>
</tr>
<tr>
<td>PQ46577</td>
<td>LISTDEF Utility control statement</td>
<td>This reduces the overhead during the UTILINIT phase when a DB2 Utility is invoked on a list of objects defined with a LISTDEF Utility control statement.</td>
</tr>
</tbody>
</table>
In Table A-4 we briefly describe the DB2 PM APARS for IRLM 2.1.

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ45176</td>
<td>BUILD2 phase</td>
<td>The parallel BUILD2 phase of REORG SHRLEVEL CHANGE or REFERENCE eliminates logging of non partitioning index logical part. Applicable to DB2 V6 as well.</td>
</tr>
<tr>
<td>PQ47582</td>
<td>Utilities</td>
<td>Correction to restart functions of RELOAD. It applies to DB2 V5 and V6 as well.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>Utilities</td>
<td>Needed after the PTF for PQ47582 to reduce increased elapsed time when dealing with many partitions. It applies to V5 and V6 as well.</td>
</tr>
<tr>
<td>PQ50332</td>
<td>DSN1COPY</td>
<td>Performance is improved by using SPEED in AMD DEFINE CLUSTER for DB2 managed data sets. It applies also to DB2 V6.</td>
</tr>
<tr>
<td>PQ51347</td>
<td>MODIFY RECOVERY</td>
<td>Poor performance during MODIFY RECOVERY and MODIFY STATISTICS utility. PTFs available for V5 and V6 as well.</td>
</tr>
<tr>
<td>PQ51844</td>
<td>Load and reorg utilities</td>
<td>This reduces the elapsed time for the catalog updates when executing LOAD LOG NO and REORG LOG NO jobs on many partitions. PTF available for DB2 V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ54608</td>
<td>TEMPLATE</td>
<td>This introduces support for variable length strings in the variables of the TEMPLATE.</td>
</tr>
<tr>
<td>PQ52412</td>
<td>Real time statistics</td>
<td>It is the preparatory APAR for real time statistics. Then you need PQ48448, PQ48859, and PQ56256</td>
</tr>
<tr>
<td>PQ52412</td>
<td>C and PLI coprocessor</td>
<td>Still open</td>
</tr>
<tr>
<td>PQ47767</td>
<td>COBOL</td>
<td>This ensures that related host variables are recognized properly. Valid for V5, V6, and V7.</td>
</tr>
<tr>
<td>PQ51078</td>
<td>COBOL</td>
<td>This reduces CPU consumption by dealing differently with host variables. PTF available for V6 as well. It requires PQ53359, PQ56704, and PQ57331 to improve CPU time.</td>
</tr>
<tr>
<td>PQ55078</td>
<td>storage</td>
<td>This helps reduce application storage below the line with large number of tasks. It applies to V6 as well.</td>
</tr>
<tr>
<td>PQ55346</td>
<td>storage</td>
<td>This improves the storage shortage cushion. It applies to V6 as well.</td>
</tr>
</tbody>
</table>

In Table A-4 we briefly describe the DB2 PM APARS for IRLM 2.1.

**Table A-4  IRLM 2.1 APARs**

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ53071</td>
<td>Better out-of-storage management</td>
<td></td>
</tr>
<tr>
<td>PQ52642</td>
<td>Storage management</td>
<td>This improves the management and the compression of IRLM extended private storage.</td>
</tr>
</tbody>
</table>

**A.4 DB2 PM V7**

In Table A-5 we briefly describe the DB2 PM APARS for DB2 V7.
Table A-5  DB2 PM V7 APARs

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ51708</td>
<td>DB2 V7 upgrade</td>
<td>This adds the display of locking conflict, SQL activity report, multiple sort and qualify of threads, general interface improvements</td>
</tr>
<tr>
<td>PQ50902</td>
<td>DB2 V7 upgrade</td>
<td>Record trace support for IFCID 217, IFCID 225. new fields for IFCID 22. Must check also II13032</td>
</tr>
<tr>
<td>PQ53246</td>
<td>Instrumentation</td>
<td>Support DBM1 storage statistics in IFCID 202 for BATCH STATISTICS, equivalent to IFCID 225</td>
</tr>
<tr>
<td></td>
<td>Facility support</td>
<td></td>
</tr>
<tr>
<td>PQ56031</td>
<td>Instrumentation</td>
<td>This corrects record trace reports with identical values for TOTAL STACK and LOCAL DYNAMIC STMT CACHE POOL</td>
</tr>
<tr>
<td></td>
<td>Facility support</td>
<td></td>
</tr>
<tr>
<td>PQ46636</td>
<td>Data sharing</td>
<td>This improves the granularity of accounting class 3 data sharing GLOBAL CONTENTION for locks by adding several new fields</td>
</tr>
<tr>
<td>PQ57168</td>
<td>Performance</td>
<td>This introduces the Performance Warehouse functions</td>
</tr>
<tr>
<td></td>
<td>Warehouse</td>
<td></td>
</tr>
</tbody>
</table>

A.5 SDK

In Table A-6 we briefly describe the SDK APARS for DB2 V7.

Table A-6  SDK APARs

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PQ52781</td>
<td>SDK 1.3.0</td>
<td>Service refresh</td>
</tr>
<tr>
<td>PQ52841</td>
<td>SDK 1.3.1</td>
<td>Upgrade</td>
</tr>
<tr>
<td>PQ50780</td>
<td>SDK 1.3.0</td>
<td>Required for JDBC 2.0 driver</td>
</tr>
<tr>
<td>PQ54336</td>
<td>SDK 1.3.1</td>
<td>For use of Persistent Reusable JVM</td>
</tr>
<tr>
<td>PQ55333</td>
<td>SDK 1.3.1</td>
<td>Service Refresh 12</td>
</tr>
</tbody>
</table>

A.6 OS/390 and z/OS APARs

In Table A-7 we briefly describe the OS/390 and z/OS APARS that are related to DB2 V7 performance.

Table A-7  OS/390 DB2 related APARs

<table>
<thead>
<tr>
<th>APAR</th>
<th>Area</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OW50528</td>
<td>Data set</td>
<td>This introduces RESET/REUSE support for VSAM extended format and striped data sets</td>
</tr>
<tr>
<td></td>
<td>management</td>
<td></td>
</tr>
<tr>
<td>OW51353</td>
<td>Data set</td>
<td>After applying OW50528, it solves a failure to open a VSAM data set having an IMBEDded index and specifying REUSE</td>
</tr>
<tr>
<td></td>
<td>management</td>
<td></td>
</tr>
<tr>
<td>OW51664</td>
<td>Data set</td>
<td>The maximum number of data sets allowed open to VSAM is increased to 65 K by APAR OW51664. This should allow DB2 to increase the correspondent DSMAX value from 32 K to 65 K.</td>
</tr>
<tr>
<td></td>
<td>management</td>
<td></td>
</tr>
<tr>
<td>OW52718</td>
<td>Data set</td>
<td>Correction for VSAM LDS striping problem</td>
</tr>
<tr>
<td></td>
<td>management</td>
<td></td>
</tr>
</tbody>
</table>
## Abbreviations and acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIX</td>
<td>Advanced Interactive eXecutive from IBM</td>
</tr>
<tr>
<td>APAR</td>
<td>authorized program analysis report</td>
</tr>
<tr>
<td>ARM</td>
<td>automatic restart manager</td>
</tr>
<tr>
<td>ASCII</td>
<td>American National Standard Code for Information Interchange</td>
</tr>
<tr>
<td>BLOB</td>
<td>binary large objects</td>
</tr>
<tr>
<td>CCA</td>
<td>client configuration assistant</td>
</tr>
<tr>
<td>CCSID</td>
<td>coded character set identifier</td>
</tr>
<tr>
<td>CD</td>
<td>compact disk</td>
</tr>
<tr>
<td>CEC</td>
<td>central electronics complex</td>
</tr>
<tr>
<td>CF</td>
<td>coupling facility</td>
</tr>
<tr>
<td>CFCC</td>
<td>coupling facility control code</td>
</tr>
<tr>
<td>CFRM</td>
<td>coupling facility resource management</td>
</tr>
<tr>
<td>CLI</td>
<td>call level interface</td>
</tr>
<tr>
<td>CLP</td>
<td>command line processor</td>
</tr>
<tr>
<td>CPU</td>
<td>central processing unit</td>
</tr>
<tr>
<td>CSA</td>
<td>common storage area</td>
</tr>
<tr>
<td>CTT</td>
<td>created temporary table</td>
</tr>
<tr>
<td>DASD</td>
<td>direct access storage device</td>
</tr>
<tr>
<td>DB2 PM</td>
<td>DB2 performance monitor</td>
</tr>
<tr>
<td>DBAT</td>
<td>database access thread</td>
</tr>
<tr>
<td>DBD</td>
<td>database descriptor</td>
</tr>
<tr>
<td>DBID</td>
<td>database identifier</td>
</tr>
<tr>
<td>DBRM</td>
<td>database request module</td>
</tr>
<tr>
<td>DCL</td>
<td>data control language</td>
</tr>
<tr>
<td>DDCS</td>
<td>distributed database connection services</td>
</tr>
<tr>
<td>DDF</td>
<td>distributed data facility</td>
</tr>
<tr>
<td>DLL</td>
<td>dynamic load library manipulation language</td>
</tr>
<tr>
<td>DML</td>
<td>data manipulation language</td>
</tr>
<tr>
<td>DNS</td>
<td>domain name server</td>
</tr>
<tr>
<td>DRDA</td>
<td>distributed relational database architecture</td>
</tr>
<tr>
<td>DSC</td>
<td>dynamic statement cache, local or global</td>
</tr>
<tr>
<td>DTT</td>
<td>declared temporary tables</td>
</tr>
<tr>
<td>EA</td>
<td>extended addressability</td>
</tr>
<tr>
<td>EBCDIC</td>
<td>extended binary coded decimal interchange code</td>
</tr>
<tr>
<td>ECS</td>
<td>enhanced catalog sharing</td>
</tr>
<tr>
<td>ECSA</td>
<td>extended common storage area</td>
</tr>
<tr>
<td>EDM</td>
<td>environment descriptor management</td>
</tr>
<tr>
<td>ERP</td>
<td>enterprise resource planning</td>
</tr>
<tr>
<td>ESA</td>
<td>Enterprise Systems Architecture</td>
</tr>
<tr>
<td>ESS</td>
<td>Enterprise Storage Server</td>
</tr>
<tr>
<td>ETR</td>
<td>external throughput rate, an elapsed time measure, focuses on system capacity</td>
</tr>
<tr>
<td>FDT</td>
<td>functional track directory</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Program</td>
</tr>
<tr>
<td>GB</td>
<td>gigabyte (1,073,741,824 bytes)</td>
</tr>
<tr>
<td>GBP</td>
<td>group buffer pool</td>
</tr>
<tr>
<td>GRS</td>
<td>global resource serialization</td>
</tr>
<tr>
<td>GUI</td>
<td>graphical user interface</td>
</tr>
<tr>
<td>HA</td>
<td>Host adapter</td>
</tr>
<tr>
<td>HFS</td>
<td>Hierarchical File System</td>
</tr>
<tr>
<td>HPJ</td>
<td>high performance Java</td>
</tr>
<tr>
<td>I/O</td>
<td>input/output</td>
</tr>
<tr>
<td>IBM</td>
<td>International Business Machines Corporation</td>
</tr>
<tr>
<td>ICF</td>
<td>integrated catalog facility</td>
</tr>
<tr>
<td>ICF</td>
<td>integrated coupling facility</td>
</tr>
<tr>
<td>ICMF</td>
<td>internal coupling migration facility</td>
</tr>
<tr>
<td>IFCID</td>
<td>instrumentation facility component identifier</td>
</tr>
<tr>
<td>IFI</td>
<td>instrumentation facility interface</td>
</tr>
<tr>
<td>IPLA</td>
<td>IBM Program Licence Agreement</td>
</tr>
<tr>
<td>IRLM</td>
<td>internal resource lock manager</td>
</tr>
<tr>
<td>IRWW</td>
<td>IBM Relational Warehouse Workload</td>
</tr>
<tr>
<td>ISPF</td>
<td>interactive system productivity facility</td>
</tr>
<tr>
<td>ISV</td>
<td>independent software vendor</td>
</tr>
<tr>
<td>ITR</td>
<td>internal throughput rate, a processor time measure, focuses on processor capacity</td>
</tr>
<tr>
<td>ITSO</td>
<td>International Technical Support Organization</td>
</tr>
<tr>
<td>IVP</td>
<td>installation verification process</td>
</tr>
<tr>
<td><strong>JDBC</strong></td>
<td>Java Database Connectivity</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------</td>
</tr>
<tr>
<td><strong>JFS</strong></td>
<td>journaled file systems</td>
</tr>
<tr>
<td><strong>JIT</strong></td>
<td>Just in time (Java compiler)</td>
</tr>
<tr>
<td><strong>JNI</strong></td>
<td>Java Virtual Machine</td>
</tr>
<tr>
<td><strong>JVM</strong></td>
<td>kilobyte (1,024 bytes)</td>
</tr>
<tr>
<td><strong>LCU</strong></td>
<td>logical control unit</td>
</tr>
<tr>
<td><strong>LOB</strong></td>
<td>large object</td>
</tr>
<tr>
<td><strong>LPAR</strong></td>
<td>Logical Partition</td>
</tr>
<tr>
<td><strong>LPL</strong></td>
<td>logical page list</td>
</tr>
<tr>
<td><strong>LRECL</strong></td>
<td>logical record length</td>
</tr>
<tr>
<td><strong>LRSN</strong></td>
<td>log record sequence number</td>
</tr>
<tr>
<td><strong>LVM</strong></td>
<td>logical volume manager</td>
</tr>
<tr>
<td><strong>MB</strong></td>
<td>megabyte (1,048,576 bytes)</td>
</tr>
<tr>
<td><strong>MSM</strong></td>
<td>Multidimensional Storage Manager</td>
</tr>
<tr>
<td><strong>NPI</strong></td>
<td>non partitioning index</td>
</tr>
<tr>
<td><strong>NVS</strong></td>
<td>Non Volatile Storage</td>
</tr>
<tr>
<td><strong>ODB</strong></td>
<td>object descriptor in DBD</td>
</tr>
<tr>
<td><strong>ODBC</strong></td>
<td>Open Data Base Connectivity</td>
</tr>
<tr>
<td><strong>OLAP</strong></td>
<td>Online Analytical Processing</td>
</tr>
<tr>
<td><strong>OS/390</strong></td>
<td>Operating System/390</td>
</tr>
<tr>
<td><strong>PAV</strong></td>
<td>Parallel Access Volume</td>
</tr>
<tr>
<td><strong>PDS</strong></td>
<td>partitioned data set</td>
</tr>
<tr>
<td><strong>PIB</strong></td>
<td>parallel index build</td>
</tr>
<tr>
<td><strong>PSID</strong></td>
<td>pageset identifier</td>
</tr>
<tr>
<td><strong>PSP</strong></td>
<td>preventive service planning</td>
</tr>
<tr>
<td><strong>PTF</strong></td>
<td>program temporary fix</td>
</tr>
<tr>
<td><strong>PUNC</strong></td>
<td>possibly uncommitted</td>
</tr>
<tr>
<td><strong>QMF</strong></td>
<td>Query Management Facility</td>
</tr>
<tr>
<td><strong>RACF</strong></td>
<td>Resource Access Control Facility</td>
</tr>
<tr>
<td><strong>RBA</strong></td>
<td>relative byte address</td>
</tr>
<tr>
<td><strong>RECFM</strong></td>
<td>record format</td>
</tr>
<tr>
<td><strong>RID</strong></td>
<td>record identifier</td>
</tr>
<tr>
<td><strong>ROT</strong></td>
<td>rule of thumb</td>
</tr>
<tr>
<td><strong>RR</strong></td>
<td>repeatable read</td>
</tr>
<tr>
<td><strong>RRS</strong></td>
<td>resource recovery services</td>
</tr>
<tr>
<td><strong>RRSAF</strong></td>
<td>resource recovery services attach facility</td>
</tr>
<tr>
<td><strong>RS</strong></td>
<td>read stability</td>
</tr>
<tr>
<td><strong>RSM</strong></td>
<td>Relational Resource Manager</td>
</tr>
<tr>
<td><strong>RTS</strong></td>
<td>real time statistics</td>
</tr>
<tr>
<td><strong>RVA</strong></td>
<td>RAMAC Virtual Array</td>
</tr>
<tr>
<td><strong>SDK</strong></td>
<td>software developers kit</td>
</tr>
<tr>
<td><strong>SMP</strong></td>
<td>System Management Interface Tool</td>
</tr>
<tr>
<td><strong>SMIT</strong></td>
<td>IBM Silicon Valley Laboratory</td>
</tr>
<tr>
<td><strong>TCB</strong></td>
<td>Task control block</td>
</tr>
<tr>
<td><strong>USS</strong></td>
<td>Unix System Services</td>
</tr>
<tr>
<td><strong>WAS</strong></td>
<td>WebSphere Application Service</td>
</tr>
<tr>
<td><strong>WLM</strong></td>
<td>Workload Manager</td>
</tr>
</tbody>
</table>
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 on ordering these publications, see “How to get IBM Redbooks” on page 121.

- DB2 Performance Expert for z/OS, SG24-6867
- DB2 for z/OS Application Programming Topics, SG24-6300
- Large Objects with DB2 for z/OS and OS/390, SG24-6571
- FICON Native Implementation and Reference Guide, SG24-6266
- DB2 for z/OS and OS/390 Version 7 Using the Utilities Suite, SG24-6289
- DB2 for z/OS and OS/390 Version 7 Performance Topics, SG24-6129
- DB2 for z/OS and OS/390 Tools for Performance Management, SG24-6508
- DB2 UDB Server for OS/390 Version 6 Technical Update, SG24-6108
- DB2 UDB for OS/390 Version 6 Performance Topics, SG24-5351
- DB2 for OS/390 Version 5 Performance Topics, SG24-2213
- DB2 for OS/390 and z/OS Powering the World’s e-business Solutions, SG24-6257
- Storage Management with DB2 for OS/390, SG24-5462

Other resources

These publications are also relevant as further information sources:

- DB2 UDB for OS/390 and z/OS Version 7 Installation Guide, GC26-9936
- DB2 UDB for OS/390 and z/OS Version 7 Command Reference, SC26-9934
- DB2 UDB for OS/390 and z/OS Version 7 Messages and Codes, GC26-9940
- DB2 UDB for OS/390 and z/OS Version 7 Utility Guide and Reference, SC26-9945
- DB2 UDB for OS/390 and z/OS Version 7 Programming Guide and Reference for Java, SC26-9932
- DB2 UDB for OS/390 and z/OS Version 7 Administration Guide, SC26-9931
- DB2 UDB for OS/390 and z/OS Version 7 Application Programming and SQL Guide, SC26-9933
- DB2 UDB for OS/390 and z/OS Version 7 Release Planning Guide, SC26-9943
- DB2 UDB for OS/390 and z/OS Version 7 SQL Reference, SC26-9944
- DB2 UDB for OS/390 and z/OS Version 7 Text Extender Administration and Programming, SC26-9948
- DB2 UDB for OS/390 and z/OS Version 7 Data Sharing: Planning and Administration, SC26-9935
- DB2 UDB for OS/390 and z/OS Version 7 Image, Audio, and Video Extenders, SC26-9947
- DB2 UDB for OS/390 and z/OS Version 7 ODBC Guide and Reference, SC26-9941
- DB2 UDB for OS/390 and z/OS Version 7 XML Extender Administration and Reference, SC26-9949
- DB2 PM Version 7 Using the Workstation Online Monitor, SC27-0859-01
- OS/390 V2R10.0 DFSMS Using Data Sets, SC26-7339
- CICS Transaction Server for z/OS V2.2 Release Guide, GC34-5983
- CICS Transaction Server for z/OS V2.2 CICS DB2 Guide, SC34-6014
- CICS Transaction Server for z/OS V2.2 Performance Guide, SC34-6009
- CICS Transaction Server for z/OS V2.1 Operations and Utilities Guide, SC34-5717
- CICS Transaction Server for z/OS V2.1 Performance Guide, SC34-5718
- CICS Transaction Server for z/OS V2.2 Performance Guide, SC34-6009
- CICS Transaction Server for z/OS V2.1 CICS DB2 Guide, SC34-5707
- CICS Transaction Server for z/OS V2.1 Application Programming Guide, SC34-5702
- CICS Transaction Server for z/OS V2.1 Problem Determination Guide, GC33-5719
- CICS DB2 Attach: Performance and Tuning presentation by Dave Raiman in CICS Technical Conference, Salt Lake City, June 2001
- Using V7 Real Time Statistics to assist DBA managing DB2 for OS/390, presentation by Jim Teng in DB2 and Business Intelligence Technical Conference, Orlando, Florida, October 2001
Referenced Web sites

These Web sites are also relevant as further information sources:

- **DB2 for z/OS and OS/390**
  http://ibm.com/software/data/db2/os390/

- **DB2 for z/OS and OS/390 Version 7 books**

- **UNICODE**

- **DB2 for z/OS and OS/390 tools**
  http://ibm.com/software/data/db2imstools/

- **SQLJ**
  http://www.sqlj.org/

- **Java on OS/390**
  http://www.ibm.com/servers/eserver/zseries/software/java

- **Jinsight**

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROMs. Click the CD-ROMs button on the Redbooks Web site for information about all the CD-ROMs offered, as well as updates and formats.
Index

Numerics
0002  27
0003  27–28
0147  28
0148  27–28
0217  26
0219  27
0220  27
0225  27
0234  27
0319  27
0334  27
31-bit addressing  18
64-bit addressing  18

A
APARs  3
application owning regions  84
auto commit  42

B
BIND options  48

C
CCSID conversion  54
channel aggregation  6
CHAR vs. VARCHAR  43
CICS
  BIND options  77
  CPU accounting  72
  guidelines in coding applications  78
  OTE  80
  thread protocol  75
  thread reuse  74
  threadsafe  80
CICS INQUIRE TASK  67
CICS interface  3
CICS Performance Analyzer  73
CICS/DB2 attachment  61
CICS/DB2 threads  61
Class 2 time  8
Class 6 trace  19
CLASSPATH  38
  collection  79
  connection context object  44
Control Center  88
COPY  13
CTHREAD  23
cube  98
data types
  mapping Java to DB2  40
DB2
  future release  19
DB2 data types  41
DB2 OLAP Server  98
  performance comparison  99
  scalability measurements  102
DB2 OLAP Server on zSeries  3
DB2 PM  19
  Performance Warehouse  29
  RECTRACE  26
DB2 PM record trace  26
DB2 PM statistics  20
DB2 PM workstation  29
DB2CONN definition  62, 66
DB2CONWT  73
DB2ENTRY  63
DB2RDYQW  73
DB2REQCT  73
DB2TRAN  84
DB2WAIT  73
DBM1 storage  18
data sharing environment  23
  non-data sharing environment  21
  single parallel query  25
DDCS  50
DFHDB2064  84
DFHSTUP  71
  -DISPLAY THREAD output  66
DSMAX  23
DSNACCOR  88
DSNC DISP STAT  68
DSNCRCT  62
DSNI037I  89
DSNTIPESS  90
DSNTIPO  90
dynamic SQL statement caching  49
dynamic statements cache  37

E
ESCON  5, 12
ESS  12
Essbase  98

F
Fibre Channel Architecture  5
FICON  2
  addressability  6
DB2 logging  9
DB2 queries  10
DB2 utilities  13
description  6
distance  6
measurement environment 6
measurement results 8
transfer rate 6

G
global contention 28
group attach with CICS 84

H
hprof 50

I
IEEE floating point 46
IFCID 0003 110
IFCID 0018 110
IFCID 0058 110
IFCID 0217 19
IFCID 0225 19
IFCID 0234 111
IFCID 0334 27
IFCIDs changed with DB2 V7 27
II04309 18
II10817 18
II12836 110, 113
intrinsic 38
IRWW 8, 21, 52, 82
iterator 45

J
Java data types 40
Java measurement results 52
Java profiler 50
Java support 2
JDBC 32
driver types 33
overview 32
reasons to use 37
JDBC and SQLJ
drivers 38
JDBC API 42
JDBC DataSource 43
JDBC/SQL trace 51
JDK 38
Jinsight 51
Just in Time compilation 50
JVM heap 48

L
LOAD 15

M
mapping data types 40
MAXDBAT 23
ms 48
mx 48

N
NVS 6

O
objective 1
OLAP
Multidimensional Storage Manager 98
Relational Storage Manager 98
open transaction environment 61, 80
OTE 61
OW50528 114, 116
OW51353 116
OW51664 116
OW52718 116

P
package 79
PAV 6
P-lock counters 27
PQ25135 110
PQ37880 111
PQ43509 112
PQ43846 111
PQ44103 110
PQ44580 110, 113
PQ44671 113
PQ45176 112, 115
PQ45184 115
PQ45186 40
PQ45526 113
PQ45820 110
PQ45820 110
PQ46577 114
PQ46636 28, 116
PQ46835 88, 115
PQ47178 113
PQ47582 110, 112, 115
PQ47767 110–111, 115
PQ47833 113
PQ47914 110
PQ47970 111
PQ47973 27, 111
PQ48031 111
PQ48126 111
PQ48306 114
PQ48383 40, 54–56
PQ48447 88, 115
PQ48448 88, 115
PQ48522 111
PQ48905 110–111, 114
PQ49121 111, 113
PQ49173 110
PQ49328 111, 113
PQ50106 112
PQ50332 112, 115
PQ50902 19, 116
PQ50781 111, 115
PQ51347 110, 112, 115
PQ51708 29, 116
PQ51765 111, 113
Index

PQ51847  39, 47, 49, 110–111, 114
PQ52142  111
PQ52412  115
PQ52642  115
PQ52841  47
PQ53070  111
PQ53071  115
PQ53246  116
PQ53565  112, 114
PQ53571  112, 114
PQ54042  114
PQ54451  111, 113
PQ54552  112, 114
PQ54580  16, 114
PQ54608  115
PQ54754  110
PQ54755  111
PQ54756  40, 53, 55, 114
PQ55078  112, 115
PQ55247  112, 114
PQ55346  112, 115
PQ55393  113
PQ55663  111, 114
PQ55682  111, 113
PQ56031  116
PQ56256  88, 115
PQ56332  113
PQ56704  111, 115
PQ57168  116
PQ57331  111, 115
PQ58420  114
PQ58820  112
PQ58843  114
PQ60839  114
PQ61458  113
PQ63490  113
PQ63825  113
prefetch  10
PREPARE  56
PRIORITY  79
protected threads  65
PROTECTNUM  65

RECOVER  14
redbook contents  2
Redbooks Web site  121
Contact us  xvii
REORG  15
RESET  116
resource definition online  62
Resource Manager Interface  61
Resource Manager Interface purge option  85
REUSE  116
RMI  61

S
SDK level  46
SET DB2CONN TCBLIMIT() command  63
Short on Storage  62
SMALLINT  42
SMF type 110  70
SORTDATA  15
SQLJ  32
  overview  34
  reasons to use  35
  specification  34
SQLJ and JDBC  35, 37
  high performance  40
START DATABASE(DSNRTSDB)  90
STATSINST  90
Storage manager pool statistics  19
storage manager pool summary  26
striping  13
SYSIBM.INDEXSPACESTATS  89
SYSIBM.TABLESPACESTATS  89
SYSREC  15

T
table scan  11
task-related user exit  61
THREADLIMIT  64
THREADWAIT  65
TPC-C  52
TRUE  61

U
UQ56153  52
UQ56531  19
UQ58284  29

V
virtual storage budget  20
virtual storage in DBM1  20
VSAM striping  16

W
WebSphere EJB Container  60
Z
z/Architecture and DB2 for z/OS  18
zSeries  19, 102
Performance measurements are ongoing during the life of each release of DB2 for OS/390. DB2 for z/OS and OS/390 Version 7 (DB2 V7 throughout this document) has introduced several enhancements in the areas of performance and availability, and other enhancements are currently being added.

Most of these enhancements and the related performance measurements implemented in the Silicon Valley Laboratory have been documented in the IBM Redbook DB2 for z/OS and OS/390 Version 7 Performance Topics, SG24-6129.

However, information on new DB2 functions and their synergy with the evolving zSeries platform is of great value for strategic investments, and more performance measurements are under way. Currently these measurements have included topics such as FICON, Java support, CICS interface, and DB2 OLAP Server.

This redbook is meant to provide an update on the new measurements that have been implemented, and to point out the performance related maintenance that has been shipped after general availability of DB2 for z/OS Version 7. The information here contained is intended to help managers and professionals understand and evaluate the applicability to their environment of these recent functions of DB2 V7.

This IBM Redbook replaces the IBM Redpaper that was made available in February 2002, with the same title, and provides more up-to-date information.