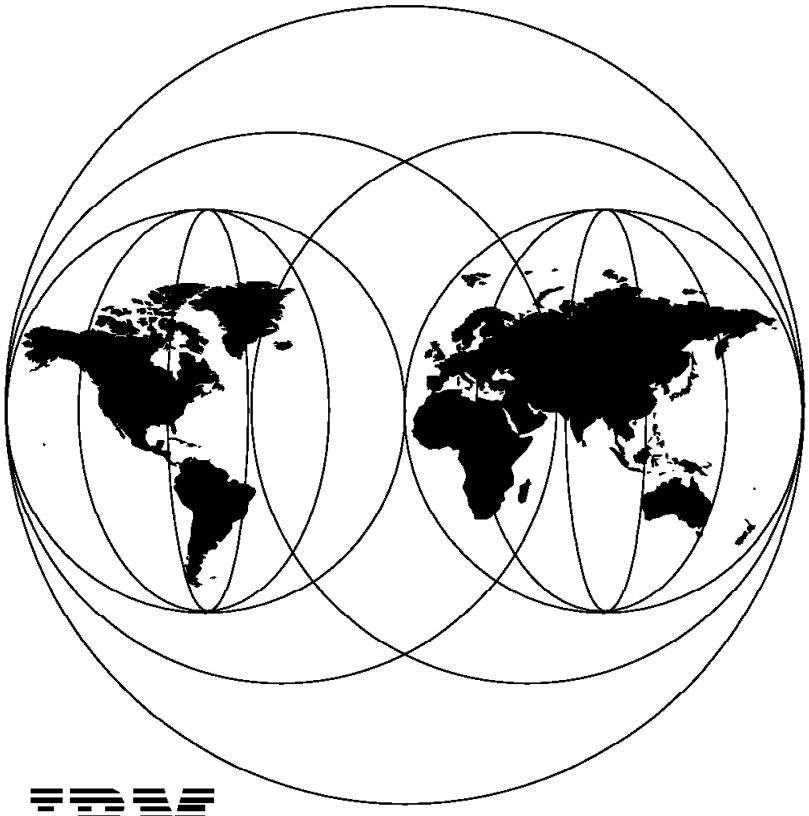
SG24-2244-00

DB2 for OS/390 Capacity Planning

December 1997



IBM

International Technical Support Organization

San Jose Center

International Technical Support Organization

SG24-2244-00

DB2 for OS/390 Capacity Planning

December 1997

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix D, “Special Notices” on page 147.

First Edition (December 1997)

This edition applies to Version 5 of IBM DATABASE 2 Server for OS/390, Program Number 5655-DB2.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. 471 Building 80-E2
650 Harry Road
San Jose, California 95120-6099

Contents

Figures

Tables

Preface

This redbook is intended for professionals to plan DB2 resources in the MVS environment. It provides a range of DB2 capacity planning techniques for online transaction processing (OLTP), query, batch, and utilities and is suitable for both new and experienced DB2 capacity planners. Estimating the resource consumption of a workload and projecting the required capacity are challenging activities, particularly in a data sharing environment. This redbook presents detailed information to help you plan and estimate the resources needed for a relational database in both data sharing and non-data-sharing environments. The wealth of information presented in this book will greatly help in a DB2 capacity planning exercise.

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.

Ravi Kumar is a Senior ITSO Specialist for DB2 for System 390 and MVS/ESA at the Application Development and Data Management ITSO Center, San Jose. He writes extensively and teaches IBM classes worldwide on all areas of DB2 for System 390 and MVS/ESA. Before joining the ITSO three years ago, Ravi worked in IBM Australia as a DB2 specialist.

Mike Bracey is a DB2 for OS/390 specialist at the Installation Support Center, Hursley, IBM United Kingdom. He has been with IBM for over 14 years and has worked in various capacities as a DB2 branch specialist, a DB2 consultant for designing, implementing, and tuning applications at many customer installations, and most recently a DB2 consultant for IBM HUON solutions. Mike is an IBM certified I/T specialist.

Shoab Kamran is a Senior Sales Specialist in Western Area Techline in San Francisco, USA. Shoab started with IBM Canada in 1969 as a programmer/analyst. He later moved to IBM USA where he has worked for 18 years as an MVS systems programmer, systems engineer, and large systems specialist. He was on assignment in IBM EMEA for 4 years as a large systems marketing specialist. His current job responsibilities include configuration and capacity planning for large systems upgrades and Parallel Sysplex.

Rolf Loeben is a Senior I/T Specialist in Germany. He has 30 years of experience in the database area and 11 years of experience at the FSC in Germany. Currently Rolf specializes in DB2 data sharing tuning. He designed the class for DB2 V4 data sharing and is coauthor of the DB2 V4 data sharing performance topics redbook.

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

- Mike Barnard — GA Life, York, United Kingdom
- Ted Blank — IBM Development, Santa Teresa
- John Campbell — DB2 Consultant, CITL, United Kingdom
- Roy Cornford — IBM United Kingdom
- Curt Cotner — IBM Development, Santa Teresa
- Rainer Eberhard — IBM Development, Santa Teresa

• Robert Gilles	IBM Development, Santa Teresa
• Harold Hall	IBM Development, Santa Teresa
• Jan Henderyckx	DB2 Consultant, Belgium
• Jeff Josten	IBM Development, Santa Teresa
• Gary King	IBM System 390 Division, Poughkeepsie
• Gopal Krishnan	IBM Development, Santa Teresa
• Bruce McNutt	IBM Storage Systems Division, San Jose
• Roger Miller	IBM Development, Santa Teresa
• Chris Mills	IBM Development, Santa Teresa
• Chris Panetta	IBM System 390 Division, Poughkeepsie
• Alison Pate	ITSO San Jose Center
• Silvio Podcameni	ITSO San Jose Center
• Bernd Saemann	IBM Germany
• Akira Shibamiya	IBM Development, Santa Teresa
• Bryan Smith	IBM Development, Santa Teresa
• Hugh Smith	IBM Development, Santa Teresa
• Jim Teng	IBM Development, Santa Teresa
• Horacio Terrizzano	IBM Development, Santa Teresa
• Henrik Thorsen	ITSO Poughkeepsie Center
• Annie Tsang	IBM Development, Santa Teresa
• Jon Youngberg	IBM Development, Santa Teresa
• Maryela Weihrauch	IBM Germany

Thanks to Maggie Cutler for editing the document. Thanks to Patricia Krohn for graphics support and editorial assistance.

Comments Welcome

We want our redbooks to be as helpful as possible. Should you have any comments about this or other redbooks, please send us a note at the following address:

redbook@vnet.ibm.com

Your comments are important to us!

Chapter 1. Introduction

This redbook deals with capacity planning for DB2 applications. We examine and estimate the computer resources needed to implement a DB2 subsystem and its applications.

The book has the following six chapters:

Chapter 1 Introduction

Chapter 2 What Is Capacity Planning?

This chapter explains capacity planning techniques and activities for DB2 in an MVS environment.

Chapter 3 DB2 Capacity Planning Methods

This chapter describes methods that can be used for capacity planning. These methods are high-level rules-of-thumb that can be used during the various stages of developing a DB2 application.

Chapter 4 Capacity Planning for DB2 Utilities

This chapter provides formulas and lookup tables.

Chapter 5 Migration to a Data Sharing Environment

This chapter provides additional capacity planning aspects to consider when migration to a DB2 data sharing environment is planned.

Chapter 6 Special Topics

This chapter provides an overview of the following topics for capacity planning:

- DB2 and RAMAC Virtual Array
- DB2 in a distributed environment

We expect the reader to have some familiarity with DB2 Performance Monitor (DB2 PM), DB2 Estimator, and DB2 data sharing terminology and concepts, besides experience with DB2 in the MVS environment.

Chapter 2. What Is Capacity Planning?

The business enterprise of the 1990s is becoming increasingly dependent on the availability of instant information from the corporate database. In the complex information processing environment of today, many factors play a role in accessing the required business data and information. When setting goals for throughput and response times are being considered, all factors have to be individually observed and analyzed for their effectiveness to meet the service levels that are necessary to achieve the corporate business objectives.

In some businesses and government departments, information technology (IT) staff go from one crisis to another, putting out fires, and have no chance to work toward the long-term goals and objectives of the enterprise. The choice is between running a business with a *management by crises* philosophy or with *management by objectives* disciplines that entail using proper capacity planning methodologies to meet the business service level objectives.

The capacity planning staff support the corporate business objectives by planning a *balanced* computer system that will avoid performance problems and not seriously impact business in the future. A *balanced* system is a system designed so that all computer resources (processor, storage, I/O) work in conjunction with each other without developing a bottleneck for any one resource. A *balanced* system optimizes the system capacity and achieves the best performance and throughput in accordance with the goals.

2.1 Why and When to Do Capacity Planning

Capacity planning is the process of estimating computer resources to meet an application′s service-level objectives. An enterprise′s computer system should accomplish more than just being simply busy. It must provide service to all its users within the requested response time and throughput objectives.

The three main reasons for having a sound and continuous capacity planning discipline established in your enterprise are:

- *No surprise for the enterprise financial management:* Even though small computer equipment has very much become a commodity purchase item, for large installations it is still a major item of expenditure. Large numbers of PCs are not inexpensive to purchase or support. Your financial management will be aware of the upgrade requirements well in advance, to properly budget for this expense.
- *Proactive management to plan future requirements*: Keeping system upgrade plans and implementation ahead of increasing workloads reaching performance thresholds optimizes your financial investments and increases IT productivity.
- *Operate a financially productive information system:* Running out of any computer resources spills over the effects on other resources, and soon the whole system starts having serious performance problems. Lack of proactive planning for future requirements may not only impact your business directly but also push you to make system upgrade decisions in haste, which could be financially more expensive and operationally less efficient.

We recommend that you make capacity planning an integral part of the application development, system enhancement, and operation processes. From the initial requirements phase to the production phase, the capacity planning process should estimate the resource requirements to satisfy user needs.

At the end of each stage of development, schedule a checkpoint review with user management and the developers. It is essential to clearly document the users′ expectations of response time, volume, and business functions. All parties must agree on any adjustments, changes, and upgrades before the next application development phase and its capacity planning is started.

You must create an assessment methodology to have a consistent measurement mechanism to ensure that the system meets your users′ expectations. You are responsible for informing management promptly if the system is unable to meet its objectives. Management then can decide to upgrade the system or reduce the functions of the application. If it is not done at the early stage of the application development process, the design may be finalized before the problems are uncovered, and it may become very costly to take any corrective action at a later stage.

2.2 System Resources and Balanced Systems

We provide below a simple description of the basic physical computer resources to have a proper level set for capacity planning discussions.

A computer system has three main physical resources:

- CPU
- Storage
  - Central
  - Expanded
- I/O
  - Channels
  - Controllers (storage directors, cache, and nonvolatile storage)
  - Devices

All subsystems and applications, eventually, allocate and/or utilize these resources.

When operating as a Parallel Sysplex or when planning to migrate to Parallel Sysplex, you must assess the impact of Parallel Sysplex on these resources as well.

Parallel Sysplex introduces the coupling facility (CF) as an additional resource, formed by:

- Links
- CPU
- Storage

Figure 1 on page 5 shows the DB2 data sharing group in a Parallel Sysplex.

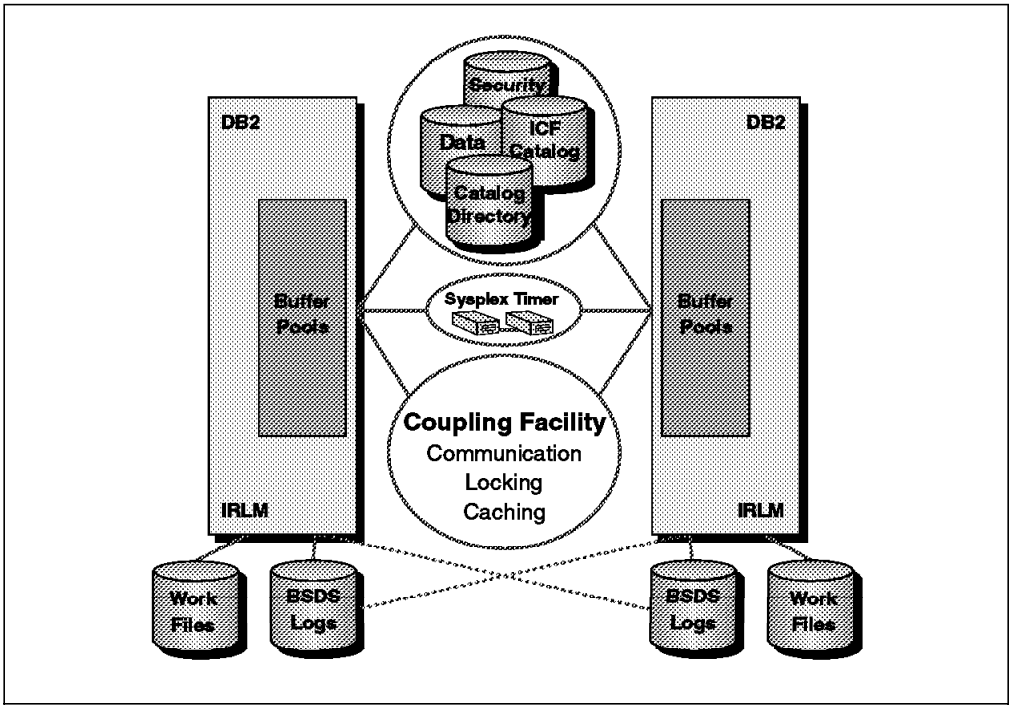


Figure 1. DB2 Data Sharing Group in a Parallel Sysplex

In a Parallel Sysplex environment, you also monitor the CF-related resources. The CF is a key resource in a Parallel Sysplex configuration. Within the CF, storage is dynamically partitioned into structures. MVS cross-system extended services (XES) services manipulate data within these structures. These structures have unique functions that enable an infrastructure suitable for workload distribution and resource balancing across multiple OS/390 (MVS) images, thereby providing a single image view of the Parallel Sysplex and preserves the integrity of shared data.

The CF is accessed through OS/390 (MVS) services executed in either synchronous or asynchronous mode. These services were introduced in MVS/ESA V5. In a capacity planning study it is important to size the resources belonging to the CF to get a *balanced* Parallel Sysplex environment.

On average, in a balanced system there should be no single bottleneck. It is extremely difficult to do capacity planning if there is a major bottleneck in your system, such as a CPU-constrained system.

If you make sure you have a balanced system, you will have the following benefits during and after the capacity planning sessions or tasks:

- Optimization of your financial investments
- Increase in IT productivity
- Meeting expectations of users by establishing service level agreements
- Avoidance of major performance problems that could seriously impact your business

2.3 Information Needed for DB2 Capacity Planning

To conduct a capacity planning study, first try to get answers to the following questions:

- Which IT resources are currently in use? (CPU, CF, devices, controllers, channels, coupling links, storage, and so forth)
- Which parts of the workloads consume most of the resources?
- What is the expected growth rate for particular workloads?
- When are the demands on current resources going to impact delivered service levels?

You must get the best possible answers to the above questions if a *balanced* system is desired as a starting point. Resources in a balanced system are not independent variables in determining system performance. The interrelationships of processing power, I/O capability, processor storage size, and network determine system performance. You can then determine the capacity on the basis of the performance objectives.

2.4 Roles of IT Professionals in Capacity Planning Process

Many different professionals are involved in the process of estimating computer resources needed by a DB2 application system, each with a different task and focus. These people need to work together.

2.4.1 Capacity Planner

The *capacity planner* mainly estimates the computer resources needed to run the system to the users′ satisfaction, meeting both the response time and volume requirements. The capacity planner has the responsibility for assessing the requirements for all computer resources. The capacity planner uses input from different sources to derive the estimates for the resources needed for the specific DB2 application. Resource estimation is initially done at a very high level, then refined at each phase of application development, to ensure that both the input for the process and the subsequent output are accurate at later stages.

2.4.2 Database Administrator

The *database administrator* is responsible for the design, maintenance, and performance of the DB2 database. The database administrator is involved in all phases of development, from data gathering (requirements phase) through moving the application into production. The database administrator uses the capacity planning methodology and formulas to design and calculate the size of the tables and the SQL that accesses them. The capacity planner incorporates the database administrator′s estimations into the overall resource estimations for the system.

2.4.3 DB2 Systems Programmer

The *DB2 systems programmer* is responsible for DB2 system performance and problem diagnosis. Monitoring the system resource utilization for buffer pools, environmental descriptor manager (EDM) pools, DB2 address spaces, log data sets, and work files are some of the important activities of this person. The systems programmer is involved at an early stage of the application

development process and works closely with the database administrator and the capacity planner for the application and system performance and capacity requirements.

2.4.4 Application Analyst

The *application analyst* has as a prime concern the cost of the application code and the SQL statements. This person works closely with the database administrator because the design of the transactions and batch jobs depends greatly on the physical design of the database. The application analyst uses capacity planning formulas to determine the cost of individual SQL statements that may be designed for batch, query, or online environments.

2.4.5 Communication Specialist

The *communication specialist* is responsible for the performance of the network that supports online applications. The network costs are generally a major part of the total cost of the system, and as such it is important to involve the communication specialist at the early stage of the development of a DB2 application.

2.4.6 Operations Staff

The *operations* staff are responsible for running DB2 applications, monitoring DB2 performance, and alerting responsible support groups immediately of any operational problem or performance bottleneck.

2.5 Capacity Planning and Application Development

Each phase of the application development cycle has different requirements for resource estimation because of the differing amounts of information available to the planner or designer at those times. In this section we discuss the use of these capacity planning techniques with respect to each phase of the application development cycle.

2.5.1 Capacity Planning Techniques

We focus here specifically on capacity planning from a DB2 perspective. We do not discuss capacity planning from the perspective of a total S/390 environment. The techniques described here address the following DB2 areas:

- Transaction processing
- Batch
- Utilities
- Query

Several techniques can be used to estimate the resource consumption for application processes and to project the capacity needed. These techniques are listed below roughly in the order of accuracy, from least accurate to most accurate.

Figure 2 on page 8 is a graphical representation of the accuracy of DB2 capacity planning techniques and the effort and cost required for each.

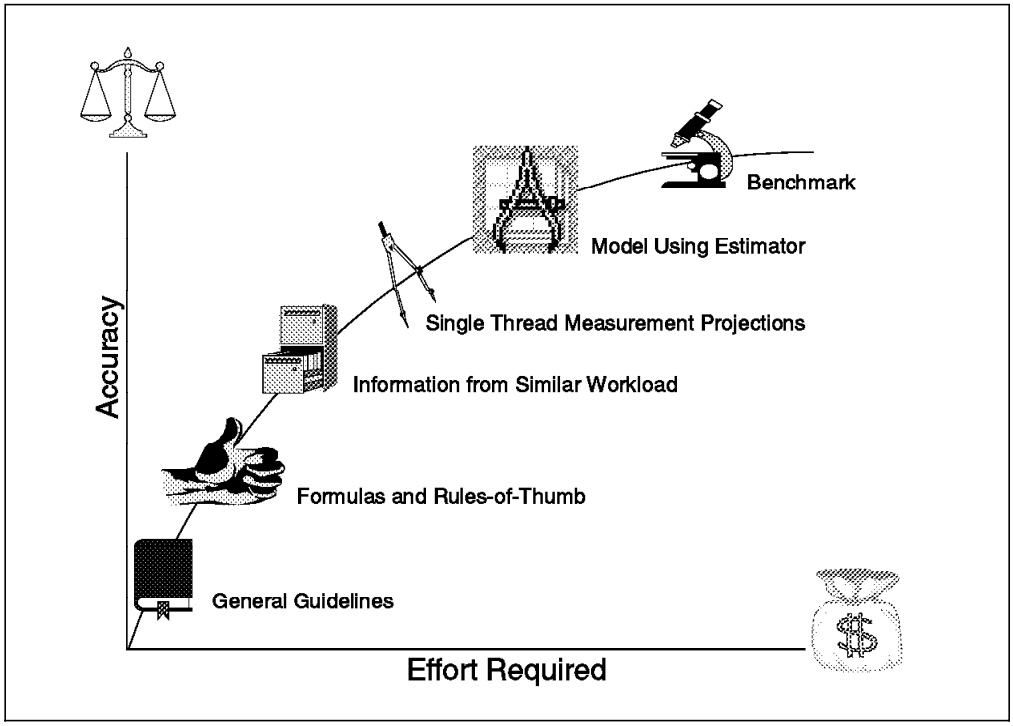


Figure 2. Capacity Planning Techniques

For each technique, the accuracy of any projected result has limitations depending on the quality of the data available and the scope of the application examined.

1. **General Guidelines** from this redbook and other related DB2 publications can be used as default values for transaction and query processing. This technique involves the least amount of work but is probably the least accurate. In practice, your workload may vary significantly from the DB2 transaction and query workloads on which these figures are based.
2. **Formulas and Rules-of-Thumb** are usually based on experience and provide a rough estimate for various workloads. You should build tables based on measurements from DB2 PM for various types of transactions, batch programs, utilities, and query users. If these types of estimates are not available in your installation, refer to Chapter 3, “DB2 Capacity Planning Methods” on page 41. We recommend that the estimates be validated and refined over time to represent ″averages″ in your particular environment. Chapter 3, “DB2 Capacity Planning Methods” on page 41 also contains basic formulas that you can use to derive slightly more accurate figures.
3. **The Information from Similar Workload** technique can be productive if the detailed information required for modeling is not available and assuming a similar application workload running in a similar environment can be found. Here are different examples of this technique:
   - Project from past experience
   - An existing DB2 application in the same environment
   - Installation with a similar type of application workload
4. **Single Thread Measurement Projections** involve loading the tables with ″productionlike″ data, taking measurements for a single user per plan, and projecting to the full workload. This technique is reasonably accurate. Even

though the work involved is significant, this technique in most cases may not give any better results than the rules-of-thumb technique. More accurate results can be obtained by using the data gathered here as input to the modeling technique.

5. **Model Using DB2 Estimator** provides more accuracy but requires somewhat detailed information. If you have a running application, then you can download the tables and SQL from the DB2 subsystem. You can use EXPLAIN and SQL trace information to set the details in DB2 Estimator.

   This technique requires both a good understanding of the application workload and a good knowledge of how DB2 processes SQL statements. Obtaining the data manipulation language (DML) and DB2 utility cost estimates from the DB2 Estimator requires less work, relative to using the DB2 Estimator to predict access path selection. The accuracy of the modeling results is very dependent on the accuracy of the input. The output from the DB2 Estimator modeling can be used as input to other modeling tools and systemwide capacity planning tools, such as the system network analysis program/simulated host overview technique (SNAP/SHOT).

6. **The Benchmark** technique with the proposed production data and SQL statements is the most accurate method that can be applied to predicting the required capacity. In practice, conducting a benchmark is an expensive and time-consuming proposition in terms of preparing the scripts, setting up the data, and tying up expensive hardware in running the tests. However, if the customer is prepared to invest the money, the results are quite accurate.

   A benchmark proves that the transactions benchmarked will run in the environment selected, at the throughput achieved. It should not be taken to apply to workloads of different sizes or different hardware or software configurations.

   Benchmarking requires many people and commitment of significant resource. For example, you may need to do the following:

   - Design and build the tables.
   - Design and write prototype programs.
   - Tailor the online customer information control system (CICS) or information management system (IMS) environment.
   - Set up a special operating environment (new software perhaps).
   - Load all the data.
   - Design the test scenarios.
   - Set up the run scripts and resources needed (CICS regions—message processing regions (MPRs)).
   - Fully commit the central processing complex (CPC) exclusively to running the tests.
   - Rerun until satisfactorily tuned with no bottlenecks.
   - Commit systems programming, database administrator, and programmer or analyst resources.
   - Commit operations resources.
   - Get assistance from the supplier.

   The relevance of the results depends on how accurately the model mirrors the production environment and the application mix that will be running

there. In summary, the accuracy of a benchmark can be high, as the workload can be measured in detail. This is rarely the case.

As an application project progresses through the stages of development, more detailed information about the DB2 workload is made available. Therefore, it is possible to refine the resource consumption estimates for the applications as they progress through the development cycles.

2.5.2 Application Development Cycle

In this section we discuss the use of the capacity planning techniques described in 2.5.1, “Capacity Planning Techniques” on page 7 with respect to each phase of the application development cycle and their relative merits in terms of workload required, accuracy, and results obtained. Each phase is examined separately.

2.5.2.1 Requirements Phase

In this phase, the amount of data available is usually minimal and the sizing broad. If the application is being built to replace an existing one, historical data on the existing application is available from measurement reports. This history, together with estimates for the new transactions to be coded, can be translated into a new estimate for resources. The estimates will therefore be fairly rough and will typically be based on experience, rules-of-thumb, and your estimation of the situation.

In other situations, however, more detailed estimates may be required. Examples are a very large application design or a tender for software and hardware for a large organization. In these instances it will be necessary to use detailed modeling tools or even benchmarks. This is the most difficult time to apply these more detailed techniques, as so little is known about the final design of the application. What is used as input data to the modeling tools is likely to have little in common with the final design. It is therefore important to reassess the estimates later, when more concrete data is available. Your capacity plan at this early stage should point out clearly the need to rerun the estimates at a later stage of development.

Given these considerations, the approaches used during the requirements phase would be:

- **General sizing**, based on some set of guidelines developed by either your database administrator or in conjunction with your IBM Product Specialist:
  - Extrapolation from an existing application
  - Comparison to similar application in a reference account
  - Rules-of-thumb from experience
  - Previously published benchmarks

  The accuracy of any general sizing approach will be fair probably only in the range of + or −100%. For individual SQL statements, where the formulas described in this book are used, the accuracy may be closer to + or −50%. However, for a complete application, the result will differ according to the amount of averaging that takes place and, more importantly, the number and accuracy of the assumptions made.
- **Modeling**, using either projected data or existing data.

If a modeling exercise is warranted at this phase, it will be necessary to collect a significant amount of data. If the data is accurate, the result can be expected to be in the range of + or −25% of the actual execution cost. The result will be entirely dependent on the accuracy of the data. The high risk of inaccurate data is a good reason to discourage modeling until a later stage of development, when more is known about the final design.

Remember, averages can be misleading. In cases where you have several transactions that have very ″light″ workload profiles, and several transactions with ″heavy″ workload profiles, your average of these two groups would be a ″medium″ workload profile for the total set of transactions. This would be misleading, as a ″medium″ transaction profile is not representative of any transaction described.

2.5.2.2 Analysis and Design Phase

In this phase, knowledge of the final application design increases significantly. The number of jobs, programs, and transactions is clearly defined, as is the amount of data manipulated by each task.

As the phase progresses, the design of the tables is finalized. The SQL for each transaction, program, or query can be produced or prototyped. Access paths can be deduced or established by using the EXPLAIN function.

A number of approaches can be taken here:

- **Paper/pencil estimates**:

  This approach involves detailed rules-of-thumb calculations.

- **Prototyping skeletons with imbedded SQL**

  This prototyping may be single-thread measurement of **SQL** statements, or it may be more exact prototyping of the transaction with, for example, **VisualAge Generator**.

- **Modeling**

  Depending on the amount of data available, this modeling can be done with DB2 Estimator or TeleProcessing Network Simulator (TPNS). DB2 Estimator provides an estimate of the resources needed based on projected volumes of data, projected catalog statistics, the number of I/Os, and other pertinent data. TPNS requires the actual or prototype transactions to be executed in a specific mix on the target, or equivalent system, at specifiable rates per second.

  DB2 Estimator gives a projected resource consumption figure, whereas TPNS gives actual execution times and throughput results. An advantage of using TPNS is that the behavior of the application is measurable by using DB2 PM. It is therefore possible to examine such things as locking contention and I/O bottlenecks and to identify poor-performing prototype transactions.

- **Benchmark**

  This approach requires a specific well-tuned environment and a defined set of transactions executed in a specific mix against a set of previously defined and measurable objectives. A tool like TPNS may be used here to generate the necessary volume of transactions from one or more workload scripts. TPNS can read input from IMS or CICS log tapes so that transaction scripts can be made to approximate the real environment. The same comments made about benchmarking in the requirements phase apply here. At this

stage, however, there can be more confidence that the designs being measured are closer to reality.

Detailed calculations using various rules-of-thumb give more accurate results in this phase than in the requirements phase, as the detailed SQL is known. Prototyping has the additional advantage of grouping a number of SQL statements into an application shell in the transaction monitor environment. This gives a more realistic result, as it includes online cost as well as the SQL cost. The cost of application logic must still be added to this result.

2.5.2.3 Development Phase

This section refers to the Produce and Build/Test phases. Information from the design phase can now be expanded and refined as the application is written or the standard application building blocks are assembled. At this point it is possible to:

- Refine estimates for each transaction, program, and individual SQL statement for a better assessment of the application logic and the access paths
- Use some actual test execution results to verify and modify the estimates
- Start detailed modeling, using a tool like DB2 Estimator, where it is not possible to build prototype tables of production size.

Testing during this phase either confirms or invalidates the estimates from the previous phases, allowing further refinement of the plan. Now that transactions are tested, actual CPU and I/O times can be obtained from test runs against sample data. They also provide another test of the rules-of-thumb being used. It is important to remember that these estimates are still sensitive to the amount of data and the content of the rows. If there is insufficient data, or the key values do not represent the cluster ratios of the production data, these measurements may still be inaccurate.

The simplest tool to use at this time for measuring the resources used is DB2 PM. The Accounting Report provides enough information to measure the resource used in and out of DB2. For the online part, use the performance statistics available through the CICS and IMS subsystems.

TPNS can be used to test whether the system can support the transaction throughput required, and SMF data can be processed through DB2 PM. This stage is also the most appropriate time to ask the IBM Product Specialist to use DB2 Estimator to model the final SQL statements. The model requires estimates of the characteristics of the production data, such as number of rows in tables and number of clustering and nonclustering indexes. The IBM Product Specialist is the most qualified person to advise you whether DB2 Estimator can provide more accurate estimates than you already have.

Of the choices available, test execution results are the most accurate, provided they are run on tables the same size as the production data. Simple selects and insert and/or update statements will vary little whether the table is small or large. Complex SQL may have joins, subselects, correlated queries, and many complicated predicates against very large tables (tables greater than 10 million rows). These SQL statements may use the expected access path on test tables, but when executed on production data with large numbers of rows, they may behave very differently, as described above.

2.5.2.4 Production Phase

Once the application is in production, it can be measured to check that it is performing as expected. The measuring tools that can be used here are:

- DB2 PM
- IBM Performance Reporter for MVS
- OS/390 Resource Measurement Facility (RMF)
- Statistics from CICS
- Statistics from IMS

Measurement of the actual application execution gives the most accurate data possible.

2.6 Capacity Planning Tools

We focus here specifically on capacity planning from a DB2 perspective. We do not intend to discuss capacity planning from the perspective of a total S/390 environment. The tools described here are those used mostly for DB2 monitoring, estimating, and modeling. For a comprehensive discussion of S/390 capacity planning tools, refer to *OS/390 MVS Parallel Sysplex Capacity Planning* (SG24-4680).

One of the most important steps in creating a capacity plan is the selection of the tools to:

- Collect data that will be used during the planning process
- Perform the estimation of resource requirements for the capacity plan

2.6.1 Availability and Selection of Tools

It is essential that the tools are used for the most appropriate purpose at the correct point in time. You should also ensure that the amount of effort that is required to obtain an accurate result with a particular tool is within your and your management's expectation.

The tools available can be divided into two basic categories: those that can be used to monitor and measure existing application information and those that can be used to estimate and model resource consumption.

Monitoring and Measuring Tools

- Visual Explain
- Statistics from CICS and IMS
- RMF
- IBM Performance Reporter for MVS. Service Level Reporter (SLR) and Enterprise Performance Data Manager (EPDM) were the predecessor products to the Performance Reporter.
- DB2 PM

Estimating and Modeling Tools

- DB2 Estimator
- SNAP/SHOT

2.6.2 Monitoring and Measuring Tools

In this section we describe the tools for monitoring and measuring the amount of resources used by a DB2 application.

2.6.2.1 Visual Explain

Visual Explain is a workstation tool. It can be used to establish which access path is being used and confirm that it is the one expected. Visual Explain is useful in all phases once an SQL statement has been written. It is essential in more detailed capacity estimation at the individual SQL statement level.

2.6.2.2 Statistics from CICS and IMS

Statistical data can be obtained for CICS and IMS subsystems that can be used as input for further analysis and resource estimation.

To monitor DB2 and CICS, you can use:

- RMF Monitor II for physical resource utilization
- Generalized trace facility (GTF) for detailed I/O monitoring when needed
- CICS monitoring facility (CMF) for performance information about each CICS transaction executed
- Performance Reporter for MVS for application processor utilization, transaction performance, and system statistics

In addition, the CICS attachment facility DSNC DISPLAY command allows any authorized CICS user to dynamically display statistical information related to thread usage and situations when all threads are busy.

To monitor DB2 and IMS, you can use:

- RMF Monitor II for physical resource utilization
- GTF for detailed I/O monitoring when needed
- IMS Performance Analysis and Reporting System (IMSPARS) or its equivalent for response-time analysis
- IMS DC monitor or its equivalent for tracking all IMS-generated requests to DB2
- Fast Path Log Analysis Utility (DBFULTA0) for performance data

In addition, the DB2 IMS attachment facility enables you to use the DB2 DISPLAY THREAD command to dynamically observe DB2 performance.

2.6.2.3 OS/390 Resource Measurement Facility

RMF reports provide information about the activities of individual jobs, specified groups of jobs, or all jobs in the system.

It also measures and reports on the activity and availability of system hardware and software resources, such as processors, devices, real storage, address spaces, serially reusable resources, and JES.

It has three monitor functions: *Monitor I*, *Monitor II*, and *Monitor III*.

Monitor I provides long-term data collection and printed reports about system workload and resource utilization. A Monitor I session runs as a background task.

Monitor II provides online measurements, on demand, for use in solving immediate problems. A Monitor II session can be regarded as a SNAP/SHOT session.

Monitor III provides short-term data collection and online reports for the continuous monitoring of system status and solving performance problems.

2.6.2.4 IBM Performance Reporter for MVS

IBM Performance Reporter for MVS, a part of SystemView for MVS, is a systems management tool that processes and transforms detailed measurement data recorded by other products such as OS/390, RMF, Virtual Telecommunication Access Method (VTAM), CICS, IMS, and OS/400. The analyzed data is stored in a DB2 database. Use of DB2 by the Performance Reporter for MVS provides for easy access to the stored performance data and makes access to distributed data across multiple platforms possible.

Performance Reporter for MVS significantly improves user productivity and ease of implementation through powerful dialogs and online help facilities. Based on the user′s definition, Performance Reporter for MVS selects which data to collect and how long to keep it.

Performance Reporter for MVS also includes a set of reports that address the user′s requirements. This systems management tool collects and transforms detailed measurement data into reports that focus on key business areas such as availability and performance management and service-level reporting.

Complex distributed computing environments require effective management and control of system resources. Performance Reporter for MVS collects system performance data, summarizes it, and saves it in a DB2 database, giving you the information you need to maintain a high level of service for your users.

With Performance Reporter for MVS, performance data can be collected from various systems on a distributed network and analyzed on an MVS system.

Performance data can be collected from the following areas:

- MVS and VM systems and their major subsystems
- Network
- CICS
- IMS
- AS/400 systems
- RS/6000 systems
- UNIX systems running HP-UX and Sun Solaris
- OS/2 systems monitored by the IBM Systems Performance Monitor/2 (SPM/2) product

Performance Reporter for MVS complements RMF, Network Performance Monitor (NPM), and other products by collecting log data from these products, enabling you to do performance, capacity, and service-level analysis based on long-term trends.

2.6.2.5 DB2 Performance Monitor

DB2 PM is a feature of DB2 for OS/390 Version 5 and is IBM′s tool for the monitoring and reporting of resource consumption in a DB2 environment. It provides both detail and summary reports for each user task as well as global subsystem statistics.

The monitoring and tuning information that is provided to the user by DB2 PM is displayed as a set of reports and graphs. Some of the areas covered by these reports are:

Accounting Detail and Summary Reports: These provide aggregates, for example, on the number of different types of SQL operations executed by a task as well as locking, suspensions, I/O, CPU, and elapsed time. Reports may be produced for specific time periods.

I/O Activity: These reports provide both summary and detail data on buffer pool activity, the EDM pool, and the active and archive logs. For example, the Buffer Pool report shows data on the I/O for each plan and authorization ID, and the time taken to complete the activity.

Locking: These reports provide data on suspension and contention in both summary and trace format.

Record Trace: Record Trace has summary, short, long, and DUMP format for each instrumentation facility component identifier (IFCID) that is being traced. Record Trace is used for problem resolution rather than for capacity planning.

Statistics: This report is the prime source of data on the DB2 subsystem itself. It provides global measurements on all activity occurring in DB2. It covers CPU and elapsed time as well as I/O, buffer pool, and logging activity, for example.

SQL Activity: SQL Activity traces and reports provide data on the amount of time spent in various components of the task executed. They provide a means of ascertaining where time has been spent.

Utility Activity: Utility Activity traces and reports provide data on the DB2 bind and utility activity for a particular thread. Information about bind, rebind, and free of DB2 packages and plans and time spent in the different phases of a utility are reported.

System Parameters: This report simply details the system parameters as invoked at startup.

DB2 PM is primarily intended to provide historical analysis of the trace information recorded at the time of execution. Through the instrumentation facility interface (IFI), users can also see snapshots of the current activity in the system. Taking snapshots during peak periods sometimes provides the capacity planner with some insight into the characteristics of that particular configuration and application workload.

DB2 PM can therefore be a useful tool in measuring the size of existing applications, either on a global basis, or on a transaction-by-transaction basis. This information can then be used to project future growth of the

existing applications or provide data that can be used to develop rules-of-thumb that reflect the characteristics of your particular environment.

For a complete description of DB2 PM refer to:

DB2 PM for OS/390 Version 5 Online Monitor User's Guide, SC26-8990

DB2 PM for OS/390 Version 5 Batch User's Guide, SC26-8991

DB2 PM for OS/390 Version 5 Report Reference, Volume 1, SC26-8985

DB2 PM for OS/390 Version 5 Report Reference, Volume 2, SC26-8986

2.6.3 Estimating and Modeling Tools

In the sections that follow we describe the tools you can use to estimate, model, or predict the amount of resources that will be used by a DB2 application.

2.6.3.1 DB2 Estimator

DB2 Estimator is a feature of DB2 for OS/390 Version 5 and is an easy-to-use tool for estimating the performance of DB2 for OS/390 Version 5 applications. Run it on your desktop personal computer, or take it with you on your laptop. Whether you want a simple table sizing or a detailed performance analysis of an entire DB2 application, DB2 Estimator is available to help you save time, lower costs, get new applications into production, or enhance and modify your existing applications.

Major Benefits: Following are the benefits and useability features of this tool:

- Provides accurate estimates of DB2 application capacity requirements
- Helps you to understand the impact of larger tables and increasing volumes of processing.
- Lets you view costs instantly online and print or export files to spreadsheet applications
- Provides online help and an easy-to-use graphical user interface
- Is used without a connection to a DB2 system
- Provides downloading of table and SQL definitions, using flat files

You can download *DB2 Estimator* free of charge, using the Internet, from **http://www.ibm.com/software/download**.

Note

http://www.software.ibm.com/data/db2 provides lots of other useful information.

DB2 Estimator is a predictive modeling tool that estimates the cost associated with the execution of DML and utilities. It is not exactly a capacity planning tool but can be used for capacity planning with some reservations, for example, for a sample access path selection function. It requires detailed information about the characteristics of the statement being modeled, to provide accurate results. The quality of the results obtained relies on the accuracy of the input data.

When to Use DB2 Estimator You can use DB2 Estimator to estimate costs associated with applications that are being developed. As with any good capacity planning tool, it must be used in an iterative manner throughout the

development process until the time when the application is capable of being measured. Once the application costs can be measured, you will typically find that measured costs are more appropriate to use in capacity planning than estimated costs.

However, with an existing application or when considering changes to an existing application, DB2 Estimator may be useful. Also when considering design changes that are tuning related, such as adding or deleting indexes, or changing the normalization of tables, DB2 Estimator can be useful. Some people have found DB2 Estimator to be occasionally useful in helping understand how DB2 is processing a query, but this is not generally considered to be capacity planning.

DB2 Estimator cannot be used until you have a vague idea of what your tables will look like. Tables can be input to DB2 Estimator by you in a generic manner with input like “there are 10 integer columns, and two character columns.” Later in the development cycle you can modify the table definitions held in DB2 Estimator as you further refine the columns that will exist in your tables. It is also possible to download existing table definitions and statistics from an existing DB2 subsystem, which can be especially useful when a new application will be using existing tables.

When you have only a definition of your tables, DB2 Estimator provides some functions, but once you have reached the point where you have an idea of the SQL that will be performed against these tables and the number of rows that will be processed, DB2 Estimator can provide a lot more value. DB2 Estimator asks you to input your SQL, so if you know what your SQL looks like you can get very good estimates. If your knowledge of what your SQL is like is somewhat vague, your estimates will not be as accurate. However, it is pretty easy to quickly input SQL and specify the columns that will be retrieved and the predicates. The actual number of columns retrieved and especially the predicates involved can affect the costs of that SQL.

At the time that you start inputting SQL to DB2 Estimator, and the number of rows that will be processed, you will also probably input the number of times that the SQL will be executed. Later in the development cycle you can further refine these values, or you may find that you are already happy with the level of estimates that have been made.

Once your applications are in production, or at least in test with a reasonable level of data in the tables, there are situations where you may also need some other tools for:

- Capacity planning: While DB2 Estimator can provide you with an estimate of the costs of an application, measurement tools like DB2 PM can provide a measured cost of an application. In general, once an application is in production, measured costs will be easier to obtain, and possibly more accurate than estimated costs. Further, if you keep a history of measured costs, you can estimate future costs based on any historical changes. For example, if your application has had the same costs as the database has doubled, you can estimate that the application will continue to have (nearly) the same costs as your database continues to grow.
- Performance problems and tuning: Using DB2 Estimator for tuning is possible and can be useful, but in most cases tuning tools such as Visual Explain and DB2 PM traces are more appropriate in determining where a problem lies.

What DB2 Estimator Can Do: When you input the definition of your table to DB2 Estimator it calculates the space required by that table and its indexes. This calculation is not too complex, and you can do it yourself with sufficient accuracy. But it is an important aspect of your capacity planning that should not be overlooked. If you do not have sufficient direct access storage device (DASD) capacity, you cannot begin to run your application.

More importantly, once you have your table definitions, you can easily start getting estimates of what the costs will be to run utilities against your data. In many shops utility processing is a very important consideration especially for those utilities that can cause normal SQL processing to be locked out. If utility costs are excessive, you may be able to use partitioning, or normalization to keep utility costs within your bounds. Doing this before you have designed SQL may keep you from having to rework SQL design.

For SQL, DB2 Estimator gives you an estimate of the CPU and I/O costs associated with executing the SQL. You must input the SQL, and then DB2 Estimator will ask you how many rows are processed as groups of predicates are applied.

These SQL cost estimates are probably the most powerful and important feature of DB2 Estimator. But in order to provide these, DB2 Estimator first must determine which access path DB2 will use. So one output from DB2 Estimator can be the estimate of the access path DB2 will use. This can be thought of as an estimate of the EXPLAIN information. In those cases where you think you know what the EXPLAIN information will be, you can also override the access path selected by DB2 Estimator to get a better cost of the SQL.

Additionally, DB2 Estimator provides functions to group SQL into transactions (or batch jobs) and include additional costs like the costs of a commit. This helps to obtain more accurate estimates of what that application will cost.

Finally, DB2 Estimator provides the ability to define a set of transactions with transaction rates to determine the total system capacity that will be required. This function in DB2 Estimator uses queuing theory modeling with the estimated application costs, but you may obtain equally valid results taking the application cost estimates from DB2 Estimator and using them in any other capacity planning methodology.

To summarize, DB2 Estimator can be useful throughout the design of an application and its database. When you input table definitions, you can obtain estimates of the amount of DASD space required, as well as the costs associated with running utilities against those tables. When you also input the SQL, you get cost estimates for that SQL, and these can be grouped together to give you cost estimates of your application, or the total impact on your system.

2.6.3.2 SNAP/SHOT

SNAP/SHOT is a discrete simulator tool developed and supported by the National Systems Services and Support (NSSS) group in IBM Raleigh. SNAP/SHOT services provide simulation of a broad range of hardware, software, and networking environments to support performance analysis, capacity planning, and design services. Originally SNAP/SHOT was designed to model a total communications system or parts of a total system. It has since been extended to model the host and I/O subsystem. The input to the SNAP/SHOT model is detailed information from system traces of the actual system execution.

As such SNAP/SHOT has little benefit for the capacity planner who is planning a new application. It is highly unlikely that the capacity planner will have the information to justify modeling using SNAP/SHOT.

The host or CPU portion of the SNAP/SHOT model simulates CPU activity based on definitions of processor type, operating system, and subsystem definitions, such as IMS, CICS, Time Sharing Option (TSO) or other application program definitions, and the work they perform. The results include machine utilizations and response times for most interactive applications.

When to Use SNAP/SHOT: SNAP/SHOT modeling of the DB2 workload relies on estimates of CPU and I/O resource consumption for each program or transaction. Therefore, the SNAP/SHOT model is best used to model the effect of all components on the total system. This would include application code, transaction managers, operating system, I/O subsystem, network response time, and database manager. Some examples of cases where SNAP/SHOT could be used are:

- Capacity planning for interactive systems, to determine:
  - The maximum number of TSO users a given processor can support based on existing TSO users in the system
  - The CPU best suited to accommodate a new online application and meet the users′ service level for response time. You can do this by using the existing applications in the system as a model
- Network design analysis for the following:
  - The value of Systems Network Architecture (SNA) relative to other transaction processing (TP) architectures
  - Response times attainable with a given network design
  - Whether a single, high-speed trunk line is better than several slower trunk lines
- The effect of new, additional, or cached devices on the performance of the I/O subsystem.

When Not to Use SNAP/SHOT: The studies listed below are not appropriate to be handled by SNAP/SHOT. Suitable tools and techniques described in 2.6, “Capacity Planning Tools” on page 13 and 2.8, “Capacity Planning Activities” on page 36 should be used for such studies.

- Capacity planning for a batch-only system
- Estimation for new application where no actual data is available from an existing system
- Real storage analysis
- Pricing for a new network design
- DASD requirements
- System tuning, as SNAP/SHOT assumes that a system being modeled is adequately tuned

Guidelines for Usage: SNAP/SHOT models are complex, requiring specific training in generating and running models. SNAP/SHOT is therefore not available directly to the customer; it is an IBM service offering. The IBM DB2

Product Specialist is the person who can best advise you whether SNAP/SHOT is an appropriate tool to consider in a particular capacity planning exercise.

2.7 DB2 Components and System Resource Requirements

The key DB2 components affecting the system resources planned for capacity and performance, to accommodate a DB2 application, fall into three categories:

- DB2 subsystem components
- DB2 functions
- DB2 applications and utilities

In this section we list the DB2 subsystem components and their default system resources requirements. These defaults are for guidance only and must be adjusted to correspond to the requirements of your own system, which will, in most cases, be significantly different. For detailed discussion of the various capacity planning factors for the DB2 application, database, and related utilities, see Chapter 3, “DB2 Capacity Planning Methods” on page 41 and Chapter 4, “Capacity Planning for DB2 Utilities” on page 71.

2.7.1 DB2 Subsystem Components

We describe below the resource requirements of those OS/390 system components that are created for or allocated on behalf of the DB2 subsystem.

2.7.1.1 DB2 Address Spaces

DB2 has five private address spaces, and one or more user address spaces (see Table 1 on page 22). These address spaces are described more fully in the *DB2 for OS/390 Version 5 Installation Guide* (GC26-8970).

System Services Address Space (DSN1MSTR): This address space handles system functions such as logging and archiving. This address space is small compared to the database services address space.

Database Services Address Space (DSN1DBM1): This is the largest DB2 address space. It processes database access requests from local users.

IRLM Address Space (IRLMPROC): With no CROSS MEMORY specified, (PC=NO), the control block structure is placed in the Extended Common Service Area (ECSA). This is limited by the amount specified in the MAXIMUM ECSA parameter. With PC=YES, the control blocks are placed in the IRLM address space. Each lock takes 250 bytes of storage and can be used to estimate storage for the control block structure.

Distributed Data Facility Address Space (DSN1DIST): This address space supports VTAM communications with other DB2 subsystems and executes database access requests on behalf of remote users. The size of this address space is negligible when compared to the size of the database services address space.

Stored Procedures Address Spaces (DSN1SPAS): These are DB2 established address spaces that provide isolated environments in which to execute stored procedures. In DB2 V4 you can have only one stored procedure address space managed by DB2. In DB2 V5 you can have multiple stored procedure address spaces managed by OS/390 workload manager (WLM). For details on

compatibility mode and goal mode stored procedures, see *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957) under “Using Workload Manager to Set Performance Objectives.”

Allied Agent Address Spaces: The allied agent address spaces are sometimes referred to as the user address spaces. They include TSO, IMS, CICS, and batch address spaces. In TSO, the DSN command requires 130 KB, and in the rest, the attach code requires approximately 36 KB of virtual storage. In all cases, the attach code must run below 16 MB of virtual storage.

Table 1 lists the DB2 address spaces showing the virtual storage required by each. Refer to Chapter 2-2, “Estimating DB2 Storage Needs,” in the *DB2 for OS/390 Version 5 Installation Guide*, GC26-8970 for details.

Table 1. DB2 Address Space Virtual Storage Requirement

Address Space	**Virtual Storage below 16 MB**	**Virtual Storage Total**
System services (DSN1MSTR)		2 - 5 MB Depends on your environment
Database services (DSN1DBM1)	1,334 KB	14 MB Depends on your environment
IRLM (IRLMPROC)		6 MB (adjustable) Depends on your environment
Distributed Data Facility (DSN1DIST)		0 KB (default) Depends on your environment
Stored Procedures (DSN1SPAS)	Variable	Depends on your environment
DB2 Allied Agent (TSO users)	130 KB (DSN command)	Depends on your environment
DB2 Allied Agent (IMS users)	36 KB (Attach code)	Depends on your environment
DB2 Allied Agent (CICS users)	5.1 KB (For RCT per address space)	MVS storage: 10.0 KB per CICS address space and 0.7 KB per thread Total storage (including CICS dynamic storage area): 43.8 KB per CICS address space and 12.5 KB per thread
DB2 Allied Agent (Batch users)		Depends on your environment

2.7.1.2 System Resources for DB2 Components

Below we discuss how system resources are affected by the DB2 components.

Common Service Area (CSA) and Extended Common Service Area (ECSA): DB2 places some of its load modules and some control blocks into CSA. Most of the space is in the ECSA. You should plan to have 2.1 MB of ECSA for DB2, and another 4 MB (more than 4 MB if you move to data sharing) for the space used by IRLM locks, if PC=NO is used. (MAXIMUM ECSA value on the IRLM installation panel DSNTIPJ if the CROSS MEMORY value is NO.)

Virtual Storage: Chapter 2-2, ″Estimating DB2 Storage Needs″ in the *DB2 for OS/390 Version 5 Installation Guide* (GC26-8970) describes the methods to calculate the virtual storage needed for a DB2 subsystem.

These calculations cover:

- Buffer pools
- EDM pool
- Data set control block storage
- Working storage
- Total main storage for each region

Real Storage: The amount of real storage needed varies greatly from system to system. However, as a guide, the amount of real storage for the buffer pool, EDM pool, and working storage should be the same as the amount of virtual storage. If there is insufficient real storage to hold the buffers, the number of buffers should be reduced. The default calculations provided for the DB2 system itself are approximately 31,714 KB of real storage and 45,110 MB of virtual storage. For more information, see the section entitled, ″Real Storage″ in Chapter 2-2 of the *DB2 for OS/390 Version 5 Installation Guide* (GC26-8970).

Table 2 shows the amount of storage (rule-of-thumb) to be added for each user.

Table 2. Additional Storage for Each User

Type of User	**Additional Real Storage (KB)**
Transaction	150
Query	400
Batch	700

The additional real storage requirement for query and batch users can vary significantly. In cases where the queries use facilities such as RID (record identifier) pools, the actual cost can be 10 to 20 times this value. If you are unsure of your users′ query patterns, use an initial figure of 1 MB per user.

Expanded Storage: Expanded storage is optional high-speed processor storage. If your DB2 subsystem is running under MVS Version 4 Release 3 or later, and the Asynchronous Data Mover Facility of MVS is installed, DB2 can use up to 8 GB of expanded storage by creating hiperpools. Virtual buffer pools hold the most frequently accessed data, and hiperpools serve as a cache for data that is accessed less frequently. The hiperpool holds only pages that have been read

into the virtual buffer pool and might have been discarded; they are kept in case they are needed again.

A hiperpool is an extension to a virtual buffer pool and must always be associated with a virtual buffer pool. Reducing the size of your virtual buffer pools and allocating hiperpools provides better control over the use of central storage and can reduce overall contention for central storage. Other than for the specification of buffer pool size, expanded storage exploitation by DB2 is automatic.

Direct Access Storage Device: A certain amount of DASD space is needed for the DB2 subsystem, irrespective of the space needed for user data. For instructions for calculating the space for these, see Chapter 2-2, "Estimating DB2 Storage Needs" in the *DB2 for OS/390 Version 5 Installation Guide* (GC26-8970). The chapter covers the DASD requirements for:

- DB2 libraries
- DB2 catalog
- DB2 directory
- Active log data sets
- Bootstrap data sets
- Temporary database
- Default database
- DUMP data set
- Communications database
- Resource limit specification table
- Archive log data sets

The amount of DASD storage required for the DB2 subsystem, disregarding the space needed for user databases, image copies, archive logs, or temporary data sets, is around 689 cylinders of 3390 or 449 MB for a small size site. An extra large DB2 system may use as many as 7,750 cylinders of 3390 or 2,390 MB of DASD. As applications are developed, the size of the catalog and directory grows and the table spaces of each can become very large.

Placement of subsystem data sets is critical to performance and availability. Data sets must be placed across a number of DASD devices and I/O channels. This is particularly important for log data sets. The DASD space required for user data is far greater than that needed for the DB2 subsystem. This must be sized accurately, and growth must be monitored regularly. Poor sizing and placement can result in multiple extents, out-of-space conditions, and poor performance. You must always consider the latest developments in DASD for DB2 storage requirements, when DB2 performance is of critical importance. The superior performance and capacity, reliability, and availability factors of RAMAC Virtual Array devices are explained in detail in *IBM RAMAC Virtual Array* (SG24-4951).

I/O: I/O is a critical resource for the DB2 subsystem and for application performance. The following activities have an impact on I/O:

- Logging

- System checkpointing
- Data definition language (DDL)—Reading and writing to the catalog and directory
- DML—Reading and writing to user table spaces
- Sequential prefetch
- Table space scans
- Sorting

For the DB2 subsystem, I/O to the logs for data changes and checkpointing can be a significant component of resource usage. This must be examined carefully. Applications with high volumes of updates will correspondingly have a high rate of logging, and read-only applications will have little or none. Where a high throughput is required, with a high percentage of columns being updated per row, the amount of data logged will be significant. In these cases, the I/O will be significant. I/O for user data can be minimized by the use of large buffers and expanded storage, and by sensible table design and SQL coding.

Network: The Distributed Data Facility (DDF) may have a major impact on the traffic on the network, in addition to normal traffic. This is due to additional remote requests for data and the amount of data that can be transmitted to the requesting application. The Block Fetch facility of DB2 is designed to minimize the impact of this data transfer on the network. Planning for this load is critical to the successful operation of the network and the applications using it. Failure to include this load may seriously impact any service level agreement. Application designs should also pay attention to this aspect of data retrieval.

CPU: The DB2 address spaces listed in 2.7.1.1, “DB2 Address Spaces” on page 21 also use a certain amount of CPU power. This is small, relative to the allied agent address space in which most of the work is done. This resource is used to carry out such functions as:

- Logging
- Establishing connections
- Checkpointing

In a DDF environment, remote users can execute tasks on your CPU for either planned transactions or adhoc queries. It is essential that this workload is carefully estimated and included in the capacity planning.

Use of the EDIT, FIELD, and VALID procedures also add to CPU usage and should be taken into consideration when estimating SQL resource usage.

If secondary authorization IDs are implemented with RACF user groups, there will be a CPU overhead involved in checking the lists if they are large. Building them is a once-only task for each user.

2.7.2 DB2 Functions

In this section we explain the key functions of DB2 that contribute to resource usage for applications. A common misconception that SQL processing is the majority of the work under DB2 has resulted in wildly inaccurate estimates. A simple rule-of-thumb is given in Table 3 on page 26 to show the distribution of work between the application and DB2 for different types of DB2 workload.

Table 3. Distribution of DB2 Workload by Type

Workload Type	Application Logic - Percent Range	DB2 Data Access - Percent Range
Query	5 - 10	90 - 95
Batch	20 - 80	20 - 80
Transaction	10 - 90	10 - 90

Table 4 lists the different DB2 traces, the cost of using each, and the impact of each on transaction time.

Table 4. DB2 Traces

Description of Trace	CPU Overhead	Remarks
Statistics trace provides information about system services and database statistics, deadlocks and timeouts, exceptional conditions, and data sharing	Up to 2%	Statistics trace classes 1, 3, and 4 are normally present
Accounting trace provides information related to application programs	Class 1: up to 3% Class 2: 5% - 20% Class 3: up to 5% for a typical case; much higher for very high suspensions	Information can be used for chargeback purposes
Audit trace provides security information about DB2 objects accessed	Up to 5%	Logs unauthorized attempts to access data
Performance trace provides information critical to performance tuning	1% - 100%	Start judiciously to gather data for performance tuning
Monitor trace records data for online monitoring	Class 1: up to 3% Class 2: 5% - 20% Class 3: up to 5% for a typical case; much higher for very high suspensions	Required for online monitoring
Global trace provides information about various DB2 events	1% - 100%	Do not start under normal circumstances; used generally only for debugging a problem

Table 5 on page 27 presents the DB2 functional components, the SQL processing cost in each component, and recommendations for component use.

Table 5 (Page 1 of 3). DB2 Components and the Cost of SQL Processing

Description	**Remarks and Recommendations**
Thread reuse: Thread reuse saves on the cost of creating a new thread of a simple transaction.	When few transactions execute frequently, set application design and system parameters to maximize thread reuse.
IMS/VS message processing regions (MPPs): In these regions a thread is created for each transaction.	High transaction frequency automatically causes thread reuse.
IMS/VS fast path (FP) regions: Dedicate each FP region to a single, high-volume transactions	Ensure high degree of thread reuse.
CICS/ESA: CICS/ESA allows you to define ENTRY threads dedicated to high-volume transactions and POOL threads allocated for general users to facilitate thread reuse	Analyze transactions for ENTRY or POOL threads and provide appropriate number of ENTRY and POOL threads.
TSO: TSO keeps a user connected to a thread for a long time.	There is no thread reuse in TSO.
Authorization checks: Authorization checks consume important resources.	You can specify that DB2 cache authorization IDs for plans or packages. Having IDs cached can greatly improve performance, especially when user IDs are reused frequently. There is a cache for each plan, and one global cache for packages.
Commit frequency: Issuing a COMMIT statement terminates the current unit of work and releases all acquired page or row locks.	Programs that contain only SELECT statements do not need to commit frequently. All other programs, including read-only programs that may contain the occasional update statements, should commit frequently. Analyze the trade-off between the resources held by a transaction and the cost of processing the COMMIT.
EDM pool and buffer pools: EDM pool and buffer pools are maintained in virtual storage for DB2 to manipulate data retrieved from DASD.	Insufficient storage in these pools results in poor systemwide performance.

Table 5 (Page 2 of 3). DB2 Components and the Cost of SQL Processing

Description	Remarks and Recommendations
EDM pool: The EDM pool stores structures that define the best access path to data (cursor tables) and database descriptors (DBDs).	Design the EDM pool to contain all the cursor tables and the DBDs actively referenced by the applications.
Database buffer pools: Database buffer pools are the most critical resources within the DB2 environment. Lack of sufficient buffer pool space results in: • Delay by not finding the page in the buffer pool and doing physical I/Os to DASD • Response time slowdown by doing more frequent writing to DASD • Inability of DB2 to use performance enhancement features like sequential prefetch	Study the buffer pool allocation and increase its size until no additional throughput is achieved or the operating system paging rate reaches unacceptable levels. You need to understand the cost trade-off, not just unacceptable paging, when increasing the size of the buffer pool allocation.
DB2 logging: DB2 logging records all changes made to data and other significant events.	Ensure DASD devices containing logs have acceptable service times. Define sufficient log partitions so the applications will not wait during log archive processing. Control the units of recovery (URs) so that they fit within the active log size. Active logs should be adequate to roll back any URs. Also, make active logs large enough so that archive log activity is not invoked too frequently.
Referential integrity: Usually, DB2 referential integrity (RI) gives better performance than application RI. However, there are cases where application RI gives better performance than DB2 RI.	Adds the following to the costs for capacity planning purposes: • Referential Integrity check for existence of a parent is calculated as an index-only access for one value • Cascade delete is calculated equivalent to the cost of a random delete or update For more design guidelines, refer to *Application Design Guidelines for High Performance* (SG24-2233)

Table 5 (Page 3 of 3). DB2 Components and the Cost of SQL Processing	
Description	**Remarks and Recommendations**
Temporary work files: DB2 temporary work files are used by functions such as sorts, subqueries (IN, ANY, ALL), and merge joins.	Define a large number of temporary work files and have them allocated on high-speed DASD. For very large sorts (more than 10 millions rows of data), consider using an external sort package, for example, DFSORT.

Capacity planning is useful when the physical design is understood. Therefore, below we discuss database design considerations, examine the effect the access path to the data has on the overall resource utilization, and present some remote data access considerations.

2.7.2.1 Database Design

There are two stages to the design of a database for an application: logical data design and physical data design. The logical data design represents all data requirements of the user in a normalized form. It is usually the output of the data modeling sessions after the data has been normalized. The logical data model is not implemented in real life. The logical model is the application′s ideal view of the data, before considering how to store this model in a computer database representation. There is very little that the capacity planner can do at this stage of application development, other than to participate in the design review sessions and begin to understand the system requirements and performance expectations.

The logical design is converted into a physical design with consideration given to the physical architecture of DB2. Access and performance requirements are taken into consideration from this point on, and it is generally at this stage that the capacity planner begins making estimations of the resource requirements. The estimates are initially based on crude rules-of-thumb but are refined as more detail about the application becomes available. The two key elements at this stage are table design and index design.

Table Design: There are certain points to keep in mind while designing DB2 tables from a resource utilization point of view. In this section we cover only some of them, namely:

- Designing large tables

  Large tables can cause a heavy load on system resources. To help improve their efficiency you can:

  - Create useful indexes—Indexes that are created with careful thought will improve the access path of your SQL statements and avoid costly table scans.
  - Use partitioning for large tables—Maintaining large tables in partitioned table spaces can limit the impact of running utility jobs (such as REORG or COPY), particularly during the batch window, as they can be reorganized or recovered partition-by-partition rather than all at once. Partitioning also allows you to place frequently used data on faster devices.

- Record sizes

  To make the most efficient use of DASD resources, there are several points to consider when designing records for DB2 tables. DB2 is designed to manage tables with a row size of 4 KB in a 4 KB page set and a row size of 32 KB in a 32 KB page set. As a general rule-of-thumb, however, it is recommended that the row size be kept to a maximum of 4 KB. Fixed-length columns are preferred to varying-length columns, as DB2 processing is optimized for fixed-length records. If you must use varying-length columns, there are two considerations: retrieval performance and update performance. For the best retrieval performance, place varying-length columns at the end of a row. For the best update performance, place the columns to be updated at the end of a row. If you use both retrieval and update operations, place the columns to be updated at the end, followed by the read-only, varying-length columns.

  It is also worth noting some hints that can avoid space wastage:

  – The default number of rows in DB2 V5 is 255. A new parameter, MAXROWS, on the CREATE TABLE statement, can be used to limit the number of rows on a page from 1 to 255. For DB2 versions previous to DB2 V5, a single 4 KB page cannot contain more than 127 rows. The exception to this rule for DB2 V3 and DB2 V4 is a table space that can have up to 255 rows, achieved by using ALTER TABLESPACE COMPRESS YES. The limit of 127 rows means that a row length of less than 24 bytes wastes space. For example, a 20 byte row leaves 4*127=504 bytes unusable per page. This can be overcome in DB2 V3 and DB2 V4 by compressing the table space and then decompressing it.

  – Space is wasted in a page that contains fixed-length, uncompressed records longer than half a page, because only one record can fit in a page.

- Free space allocation

  Tables that are subject to frequent insert and update activity can benefit from allocation of free space. The amount of free space must be balanced against the performance of read access to the table. In cases where large free space is defined, the number of rows per page decreases. More I/Os are then required to access the same number of rows with a smaller free space allocation.

Index Design: The benefits of indexes are obvious, in that they:

- Provide direct pointers to the required data in the table
- Eliminate a sort, if the requested order matches the index

While conducting a capacity planning study, and participating in the design of tables and indexes, remember that, although the indexes can improve the access path, the cost of maintaining the indexes can be significant if the system under consideration performs many INSERT, UPDATE, and DELETE operations on the data.

For guidelines on how to establish indexes, refer to Chapter 2-5, ″Designing Indexes,″ in the *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957). For information about determining index efficiency, refer to Chapter 5-3, ″Improving Response Time and Throughput,″ in the same manual. The *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957) provides a detailed discussion on access paths and index utilization.

2.7.2.2 Data Access

The discussion in the chapter to this point has focused on the effects that DB2 system components and the physical design of tables have on the resource consumption of the application.

In this section we examine the effect that the access path to the data has on the overall resource utilization. Specifically, we look at the components of DB2 that participate in data access, describing the various ways that DB2 can access required data.

The way in which DB2 accesses data depends greatly on how well the DB2 components were designed and on the design of the SQL statement. When writing SQL, certain facts must be kept in mind to ensure optimal resource utilization.

The DB2 component that determines how to access data efficiently is called the *Optimizer*, and the process it performs in selecting the optimal path is known as *access path selection*. It is not necessary to know how the Optimizer chooses an access path in order to use DB2 products. However, a certain level of understanding as to how the Optimizer makes its decision can be useful to the capacity planner and the database administrator. Before discussing the different types of access paths, we describe the two stages of data evaluation and qualification. Stage 1 is responsible for DB2 data management. It can be thought of as a high-level access method used by Stage 2 and the DB2 utilities on behalf of the SQL user. It is responsible for requesting data from DASD and evaluating basic predicates. This results in a reduced number of rows passed to the next stage.

Stage 2 is responsible for applying the more complex predicates in the SQL statement and returning the result to the application program or query user.

Below we describe the techniques available to the Optimizer to access data in the most efficient manner. The terminology we use is used in the formulas in Chapter 3, “DB2 Capacity Planning Methods” on page 41 and Chapter 4, “Capacity Planning for DB2 Utilities” on page 71.

Predicates: Predicates are portions of the SQL statement in the WHERE, HAVING, and ON clauses. They contain the search arguments that qualify the rows satisfying the user’s request.

The different types of predicates and their effects on performance are:

Indexable

Indexable predicates are those that can be evaluated through an index.

Stage 1 and Stage 2 Predicates

Rows retrieved by DB2 pass through the two stages of qualification described above. Those predicates that are applied in Stage 1 are called *Stage 1 predicates* (sometimes called *sargable*); the remaining predicates are called *Stage 2 predicates* (sometimes called *nonsargable* or *residual*). It is important to note that the formulas described later in this book do not support Stage 2 predicates. The DB2 Estimator is the only analysis tool that can accurately assess the cost of Stage 2 predicates.

As the cost of evaluating rows in Stage 1 is significantly lower than in Stage 2, every effort should be made in the design process to have the predicates evaluated as early as possible.

Unique Index with Matching Value: This is an efficient method of accessing the data. In this case, DB2 reads only as much of the index as needed to locate one entry that points directly to the required data page.

Index-Only Access: For index-only access, DB2 only has to process the index to evaluate the SQL predicate and return the result to the user. If all the data that DB2 needs to retrieve is in the index, it is not necessary to read any data pages. This results in a small number of I/Os and a high level of efficiency. Even if it is a nonmatching index, there is some advantage in scanning the entire index rather than the entire table.

Multiple Index Access: There are many times when the performance of an SQL statement could benefit from the use of more than one index to arrive at the result. This process is achieved by evaluating and comparing predicates through the index values, which results in a smaller set of data pages that must be accessed to return the result.

To use the multiple index access path, each predicate in a query must be a Stage 1 predicate and must reference a separate index. Multiple index access is available for SELECT, UPDATE, and DELETE statements. Multiple index access is especially useful with large tables, where the processing time for sorting and merging index entries is less than the time required to access many data pages and read many rows.

Index Scan: Performance characteristics are very different depending on whether the indexes are clustered or not.

- If the index is clustered, the data rows are stored in the sequence of the index key.
- If the index is not clustered, the data rows are not stored in the sequence of the index key.

When data is retrieved through an index, there are many benefits to the selection of a clustering index, as there is a high likelihood of many rows being retrieved in one page. Conversely, when a nonclustering index is used, the worst case can result in an I/O to a data page for every index entry that qualifies.

Table Space Scan: Table space scans may be expensive from a resource utilization point of view. DB2 must read every row of the table space to determine whether its value matches the given value. If there is more than one table in the table space, DB2 may have to process rows from all the tables in the table space, depending on whether they are segmented or nonsegmented. The scanning is different for each table space type:

- *Nonsegmented table space*

  DB2 processes every row in a nonsegmented table space. In elapsed time calculations, the number of rows is less important than the number of pages in the table space. Each page must be read. Remember, however, that the total of all pages includes data for other tables in the same table space, space left free, and space not yet reclaimed after deleting data.

- *Segmented table space*

  If the table space is segmented, DB2 reads only pages containing the rows of that table. Even if there is only one table in that table space, a segmented table space provides advantages as DB2 locates and ignores empty segments and segments containing only deleted data. This reduces the number of I/Os.

Prefetch: Prefetch is the process of anticipating the requests for data pages in advance and performing the I/O to place the page in the buffer pool, before the application requests the page. It is performed asynchronously with the originating application program and results in significant improvements to elapsed time.

Three prefetch mechanisms are implemented in DB2:

- *Sequential prefetch*

  At bind time, if DB2 detects that the user is requesting pages sequentially from the table or index, the sequential prefetch mechanism is invoked. It schedules requests for the data or index pages in advance of the application′s actual request for the pages, thereby reducing the elapsed time. Remember, however, that sequential prefetch is not invoked when the application issues many single SELECT INTO statements, each of which accesses only one row, even if the rows retrieved are in clustering sequence.

  Performance and resource utilization can be significantly improved by using a cached DASD control unit (like the IBM 3990 Model 6 Storage Control). In cases when the sequential prefetch is invoked, the storage control unit can read in and cache a full track at a time, giving a lower average I/O elapsed time when most or all pages in the track are needed.

- *List prefetch*

  List prefetch is an extension of the basic sequential prefetch and allows DB2 to build lists of data pages that are ordered in data page sequence, to be retrieved with the sequential prefetch method. This method of invoking sequential prefetch improves the performance of access using nonclustering indexes and is also used for the multiple index access.

- *Sequential detection at execution time (dynamic prefetch)*

  If DB2 does not choose prefetch at bind time, it can sometimes use it at execution time. The method is called *sequential detection*.

  DB2 can use sequential detection for both index leaf pages and data pages. It is most commonly used on the inner table of a nested loop join, if the data is accessed sequentially.

  If a table is accessed repeatedly using the same statement (for example, DELETE in a do-while loop), the data or index leaf pages of the table can be accessed sequentially. This is common in a batch processing environment. Sequential detection can then be used if access is through:

  – SELECT or FETCH statements

  – UPDATE and DELETE statements

  – INSERT statements when existing data pages are accessed sequentially

DB2 can use sequential detection if it did not choose sequential prefetch at bind time because of an inaccurate estimate of the number of pages to be accessed.

Sequential detection is not used for an SQL statement that is subject to referential constraints.

For additional details, see Chapter 5 in the *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957).

Joins: The process of combining rows of one table with rows of another table based on one or more common columns is called a *join*. The join operation is considered to be one of the major advantages of the relational database; however, it is still an expensive operation.

To join two or more tables in a single query, DB2 chooses the most efficient of the following three methods:

- *Nested loop join*

  In this join, the outer table is scanned once, to apply any predicates that relate only to that table and would reduce the number of rows that must be considered for a join. The inner table is then scanned once for each qualifying row in the outer table, to look for join candidates. The nested loop join is most efficient when the inner table has an efficient access path on the join predicate, or when only a few rows of the outer table remain after applying the local predicates.

- *Merge scan join*

  In cases where the tables are not in the join column order, the tables are sorted on the join columns. Then the inner table is scanned once for each qualifying row in the outer table. The scan of the inner table ceases when the column value exceeds the value of the column in the outer table for each scan.

  This join technique is used if the number of rows in each table that qualify for join after local predicates are applied is a high proportion of the actual rows in the table.

- *Hybrid join*

  Hybrid join uses list prefetch more efficiently than nested loop join, especially if there are indexes on the join predicates with low cluster ratios. It also processes duplicates more efficiently because the inner table is scanned only once for each set of duplicate values in the join column of the outer table. If the index on the inner table is highly clustered, the intermediate table is not materialized and not sorted.

  This join is often used if:

  – A nonclustered index or indexes are used on the join columns of the inner table.

  – There are duplicate qualifying rows in the outer table.

2.7.2.3 Remote Data Access

Access to remote data places additional load on both the requesting and serving sites. The total cost combining the cost from both sites is generally higher than the cost of executing the statement locally. Remote data access also places additional load on the communication functions.

For a more complete discussion of the relative costs of processing SQL data services, refer to the *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957).

2.7.3 DB2 Applications and Utilities

In this section we identify and describe the resources used by DB2 applications and utilities that you should include in capacity planning.

2.7.3.1 DB2 Application Components

For an application, CPU, I/O, DASD, and storage are all critical factors. The requirement for them depends on the application′s characteristics. If it is an online transaction, response time may be important, whereas for a batch program, I/O may be the critical factor.

The cost of any transaction can be divided into the following parts:

- DB2 system cost
  - Thread create or terminate
  - Transaction commit
  - Logging
- Data access cost
  - SQL execution
- Application logic cost
  - Data manipulation and screen handling
- Transaction manager cost
  - Number of threads (users)—maximum and concurrent
  - Number of CICS regions
  - Number of MPRs
  - Queuing and scheduling
  - Logging

The impact of these components is discussed in detail in *Application Design Guidelines for High Performance* (SG24-2233).

The resource requirement will vary depending on whether the application is online, batch, or query.

In an online environment, the throughput required, transactions per second, and response time needed will probably be the major factors in the total CPU needed. In a batch task, the number of updates to be done and the batch window may be the major factors. In a query environment, the major factors may be the number of users, the amount of data being retrieved, and the amount of sorting required.

In all cases, the application logic is typically about 70% or more of the total application pathlength.

The CPU, storage, and I/O for transaction, batch, and query are discussed in Chapter 3, “DB2 Capacity Planning Methods” on page 41.

2.7.3.2 DB2 Utilities

DB2 utilities use the same set of resources as applications, namely:

- CPU
- I/O
- DASD
- Storage

One may well ask why utilities have been separated for special attention. The reason is that utilities cut across both areas and may account for a large amount of resource usage. After all, they are executed for every application and for DB2 subsystem backup every day in most installations.

For DB2, they are required to back up the catalog and directory table spaces on a daily basis. This is not usually a large job but is a constant amount to be included in the capacity planning. For user data, backup and reorganization are the most regularly used, and the cost must be estimated for each application. The cost of execution is largely dependent on the size of the tables. Obviously, the greater the number of rows and columns, the more CPU needed to load, back up, and reorganize the table spaces.

If an error occurs, recovery can be a major exercise if the tables are large. You must include some estimates to allow for this eventuality. Care must also be taken to ensure that there is capacity to run both the application and data recovery. Also, you must take into account the resources and time needed for application reprocessing and the recovery of the DB2 system data in a timeframe that allows resumption of processing with minimal impact on the business. Nowadays, it is common for online shops to expect above 99.8% availability for unplanned outages. So, if it takes longer to recover from an outage than your online business system can sustain, the situation will be not acceptable for the enterprise.

2.8 Capacity Planning Activities

Capacity planning is a subset of the systems management discipline known as *performance management*. Performance management is the activity that monitors and allocates data processing resources to applications according to *service level agreements* or equivalent. In this section we discuss the steps you have to carry out to do capacity planning.

Capacity planning activities include and are linked to the other processes of performance management and systems management disciplines. Before you can conduct a capacity planning study, you have to define your activities and objectives. The major activities, keyed to Figure 3 on page 37, are:

1. Select capacity planning technique
2. Measure current workload
3. Check for a balanced system
4. Predict the future

5 Model and interpret data
6 Collect and document the capacity planning results
7 Communicate the capacity planning results

Depending on the selected capacity planning method and tool, the number of capacity planning runs per year can span between 1 and 12. The capacity planning period should not exceed 24 months, unless there are very good reasons, such as a stable workload.

Between consecutive capacity planning runs, complete reporting is done. This ensures adequate time to adapt to exceptional changes or deviations from the forecast. Figure 3 provides a sequential view of the activities to be followed for an effective capacity planning study. Except for the initial application development phase (see 2.5.2.1, “Requirements Phase” on page 10), where all activities may not be feasible, the capacity planning activities in Figure 3 must be repeated at each application development cycle.

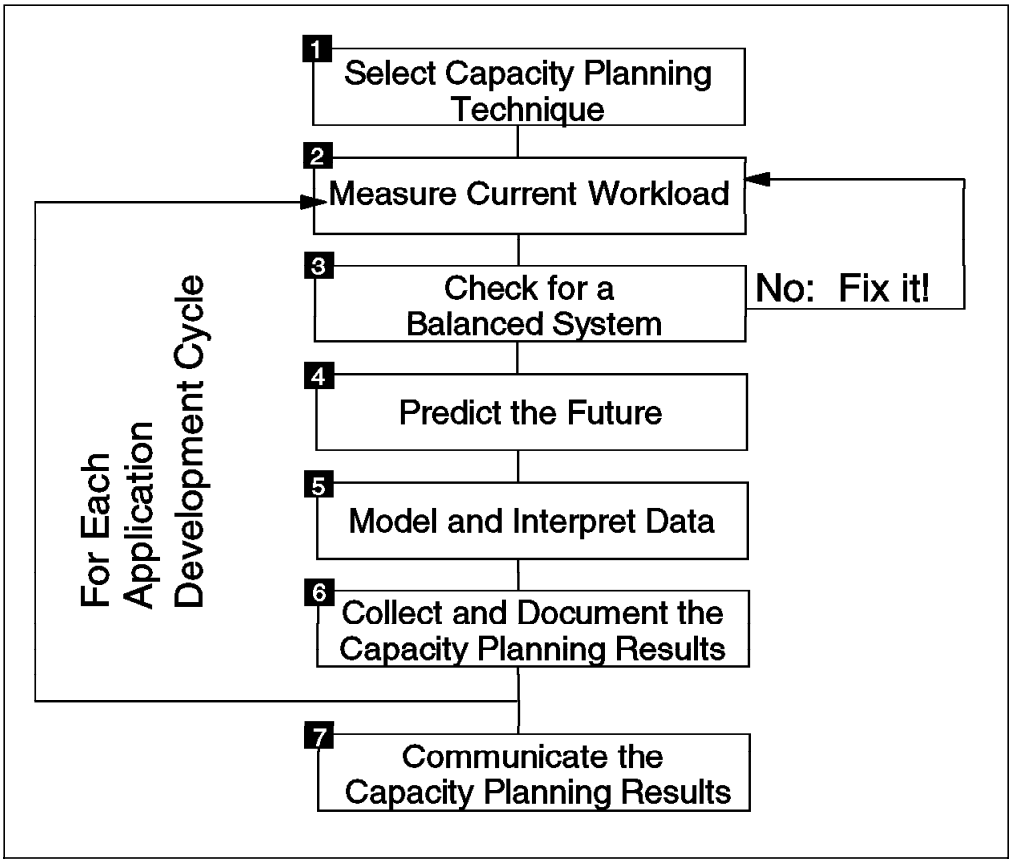


Figure 3. Capacity Planning: Step by Step Activities

2.8.1 Select Capacity Planning Technique

Based on accuracy of the projections, the first step in a capacity planning study is the choice of technique. DB2-related capacity planning techniques are outlined in 2.5.1, “Capacity Planning Techniques” on page 7. According to the selected choice, you collect different data from multiple sources of the installation.

2.8.2 Measure Current Workload

Standard performance measurement tools are typically used to capture resource usage that is currently required for the workload. The measurements should reflect the activity by unit of work (transaction, batch job, and so forth). Transactions can be grouped at a business workload level as projections are often discussed at that level.

The frequency of capacity planning involves consideration of:

- Time for the analysis
- Time for approving the investments
- Time to carry out

There also should be an understanding of the workload that cannot be delayed—workload that the business depends on or is essential for the daily business.

Often, a business has *peaks* in the system workload at certain points in time. This depends very much on the business. A bank, for example, might experience a *peak hour* every day between 3 p.m. and 4 p.m. It might also experience a *peak day* every Monday because of long opening hours at the branches. It may also experience *peak weeks* every last week in the month because of salary and other monetary transfers. The bank also knows that every December it experiences a higher workload than that of other months.

Be sure to take into account continuous availability aspects. It may be wise to overconfigure parts of the system, just in case part of the system must be taken down for service on a regular basis, primarily because of planned maintenance but also because of other outages.

Contingency allowances should also be applied. In a steady-state environment, this allowance is very small and grows with the fluctuation of the workload.

2.8.3 Check for a Balanced System

Perform an analysis of the captured data to ensure data validity. At this point, check whether you have a balanced system or not. Serious out-of-balance (overutilized) symptoms might appear, such as service level agreements that are no longer met or parts of the systems showing stress symptoms. WLM can help you use more of the existing capacity, while still managing to meet service level agreements. For details, see *Selecting a Server: The Value of S/390* (SG24-4812).

For out-of-balance systems, latent demand might be a problem when trying to estimate the workload. For example, CICS should not be going short on storage; IMS regular message region occupancy should be less than, say, 65%; average DASD response time should not exceed, say, 10 ms; and the OS/390 (MVS/ESA) capture ratio (CR) should not be below, say, 0.85. Care should be taken at this point. The best solution is probably to fix the problem and then rerun the measurements on your system.

2.8.4 Predict the Future

Now comes the hard part: assessing the future. It is very important to know the growth rate of your business and how to translate this into growth in resources. At this point, you must decide for which period you want to plan. Planning periods often go from 12 to 24 months. It is not advisable to expand the planning period because the impact of unexpected business decisions is too high. This causes a reduction of planning accuracy and therefore a reduction in the validity of the study.

With the decision to run "what if" sessions, the time comes when all the right people are consulted and asked questions such as:

- What are the company plans and goals?
- How many people will work in department XYZ next year?
- Will you change critical business processes in the future?
- How are you converting business processes to capacity planning input?
- Are you changing the IT strategy?
- Are you migrating to another IT architecture?
- Are any business mergers taking place soon?
- Do you plan to develop or buy a new application?
- Do you plan access to the World Wide Web?
- Are you implementing more client/server based applications?
- Are the users changing their usage of the system?

The new or changed business processes have to be converted and translated in terms of transactions, associated CPU, I/O, and storage resources required to support them, in terms of the CPC and the CF. Even though this is the tough part, it is also the fun part, as you get to know your business very well. If this part of the capacity planning study is not carried out with the proper level of detail, you might get very uncertain (or even wrong) predictions. Remember that well-defined and implemented systems management disciplines increase the accuracy of the predictions. *The old adage, garbage in, garbage out, still holds true.*

2.8.5 Model and Interpret Data

Now you have to model the future workload to predict how it runs on various software and hardware configurations. Several growth scenarios are often a product of this step. Several techniques can be applied in this step. Depending on accuracy, detail, invested time, money, and available skills, the appropriate method and tool are chosen. See also 2.8.1, "Select Capacity Planning Technique" on page 37 for a discussion of the relationship of effort, cost, and accuracy for any DB2 capacity planning exercise.

2.8.6 Collect and Document the Capacity Planning Results

The last step is to *document* the results of the capacity planning study. A report showing the effect of various growth scenarios addressing the business objectives is generated. Now, it is a good idea to show the relationship between the last time's prediction and the actual workload usage at the point of measurement. Probably now (or the next time around) it is a good idea to provide *feedback* to the people who provided information about the future: how accurate were their predictions? Did the business evolve the way it was planned? The feedback might ensure a more accurate prediction next time and motivate the information providers so they can see the value of their information.

To do sound capacity planning, the steps must be repeated over and over again. How often depends on the details provided and the characteristics of the business. One or two times a year is a fair rule-of-thumb. When doing capacity planning as a continuous process, most installations find it easier and easier as they get more and more experienced.

The result of the capacity planning study outlines both vertical and horizontal growth scenarios.

2.8.7 Communicate the Capacity Planning Results

Running successfully through all of the previous activities is worthless if the results are not communicated to the responsible people. To communicate the results, establishing communication links to the management, financial, and operating people is highly recommended. This also makes life a lot easier the next time around.

Chapter 3. DB2 Capacity Planning Methods

In this chapter we describe a simple approach to capacity planning together with some methods and rules-of-thumb that you can use for DB2 resource estimation. The essence of the approach to capacity planning is the establishment of a relationship between some measure of the business transactions, such as product orders, and the application transactions that are triggered by the business transaction. Without an understanding of this relationship, future capacity for an existing application can be forecast only on the basis of historical trends, which though useful, can be misleading.

The estimation methods and guidelines are most appropriate during the initial stages of application development when precise information is not available about the application design. See Chapter 5, “Migration to a Data Sharing Environment” on page 89 for data sharing considerations.

As discussed in Chapter 2, “What Is Capacity Planning?” on page 3, the primary purpose of capacity planning is to estimate the hardware resources required to operate an application at a required level of service. The most important resources are:

- Number, power, and configuration of the CPUs
- Main and expanded storage
- Size and I/O rate of the storage subsystem

In the sections that follow we present some guidelines and rules-of-thumb for each of these resources.

A Friendly Word of Warning

Rules-of-thumb can seriously damage your health.

Always attempt to verify these guidelines in your own environment. Rules-of-thumb can gain almost mythological status, at which point they are potentially more harmful than useful. To put it another way “There are no Golden Rules in this game.”

3.1 Drivers for Capacity Planning

The need for DB2 capacity planning is driven by a plan to change the production applications of an installation. Potential changes are:

- Development of a new application
- Major change to an existing application
- Migration of an application to a new database management system (DBMS)

A change to an existing application is typically either a change in the usage due to growth or an enhancement to the application function. We recommend that you use the approach described in this chapter, whatever the cause of the change. If you are doing capacity planning for an existing application, you can use measurement tools, such as DB2 PM and RMF, to measure precisely the current resource utilization rather than estimating from first principles.

If you are migrating to DB2 from a different DBMS, you will have available detailed information about the current database design and utilization. You can use this information to feed into the models described in this book. Choose the appropriate model depending on the time available for the capacity planning exercise and the level of accuracy required. Refer to *Planning for Conversion to DB2: Method and Practice* (GG24-4445) for further information. IBM internal users can also reference CONV2DB2 on MKTTOOLS.

In this chapter, we focus on the translation of a business plan to a capacity plan for the first case, development of a new application where little information is known about the application, as this is the area of greatest concern to most installations. The approach is also very relevant for an existing application.

3.2 Methodology

In this section, we develop a simple approach to capacity planning based on the business plan and using analytical methods. Using these methods, you will build a model of the resource utilization of the application. The term *model* does not necessarily imply the use of the sophisticated mathematical techniques derived from statistical or probability theory. All that is required is an understanding of simple arithmetic.

We recommend that you consider the use of a spreadsheet to simplify the drudge part of the process, which is performing the calculations, leaving you the time to concentrate on obtaining the input data, developing the model, and validating the results. The model you build will be different from any other; however there are a number of common dimensions:

- Throughput
- Design
- Technical environment

3.2.1 Throughput

Application throughput is the critical input to any capacity planning model. You may be given planning data such as transaction rates or batch records to be processed, but more often than not you will have to derive this information from a business plan. If a business plan is not available, you can only predict the future based on an analysis of historical trends. For example, if the growth in the application transaction rate has been 10% over the last year, that may be the best estimate for the coming year, given no other information. This assumes that current service levels are being met, as additional capacity may be required for either additional throughput or for maintaining acceptable response times during periods of peak utilization.

3.2.1.1 Business Plan

The business plan for the application not only defines the business objectives, scope, and function of the application but also the planned volume of business to be transacted. We recommend that you obtain from the business managers the primary business transactions and their planned usage level. Attempt to identify the high volume and high work content processes and consequent business transactions. For many applications the 80:20 rule applies, that is, 20% of the transactions generate 80% of the load.

Ideally the business plan will define the business transactions through time, typically by month or quarter. You can then calculate the growth rate.

Alternatively you might be given the business transactions for the first year and then cumulative percentage growth rates for subsequent years. Either way the data can be used to form a simple model of the capacity required.

For example, an insurance company may have planned the number of quotes, new policies, midterm adjustments, claims, and renewals for a new line of business over the first four quarters. Each of these primary business transactions will generate a number of online, batch, and query application transactions. There may also be secondary business transactions that can be derived from the primary business transactions. For example, if only the number of new policies is planned, the business managers may have a rule-of-thumb which says that each active policy will generate two midterm adjustments per year. (A midterm adjustment is simply a policy change like purchasing a new car or moving house.)

For an existing application, measure the actual business transactions achieved over a period of time so that the relationship to the actual application transactions can be established. Note that the measurement of business transactions within an organization can depend on your point of view. For example, if one person calls an insurance company on three separate occasions and on each occasion is given three quotes, how many quotes have there been? The marketing department would probably count one, as only one person has responded to the advertising campaign. The call center would count three, as they would be monitoring the number of calls which determines the need for telephony agents. The systems department would count nine, as that is what generates the application transactions and the subsequent load on the CPC.

Whatever measure of the current business transaction rate you use, ensure that it is compatible with the forecast rate being produced by the business planners. If there are a substantial number of application transactions, categorize them by high-level business function, such as insurance quotation and claims processing. One simple way to achieve this is to use the IBM Performance Reporter for MVS, which has a default DB2 table for grouping application transactions into primary business functions.

3.2.1.2 Users

Obtain information on the number and type of users. By type we mean whether they are casual users, event driven, as in a telephone call center, or heads down processing as in a traditional data preparation department. You may be able to get the number of users by department, which you can relate to the business transactions planned for that department.

You can calculate an upper bound for transaction rates by assuming a value for the think time (the time between each transaction) for each type of user. Think times can be as low as 10 sec for intensive data input processing but would typically be between 15 and 30 sec in peak periods. For example, with a think time of 15 sec, 300 users generate 20 application transactions per second.

Simple Transaction Rate Formula

$$\text{Transaction rate per second} = \frac{\text{Number of users}}{\text{Think time (seconds)}}$$

For some applications, for example, those serving the Internet, the population of users may be completely unknown and unpredictable. Capacity planning in this

instance is turned around, and, instead of calculating the capacity required for a given throughput, the maximum throughput is calculated for a given hardware capacity.

For an existing application with many business functions and a large number of users, say more than a few hundred, categorize the users into groups by business function. You can then relate the business transactions to the primary group responsible for executing those transactions. As with the business plan, the simplest way to do this is to use the IBM Performance Reporter for MVS and establish a table relating users to business function groups. If you are using IBM Resource Access Control Facility (RACF), you can run a standard extract from the RACF database, which is then loaded to DB2 tables. If a suitable RACF structure already exists, this may help with the allocation of users, and hence application transactions, into business function groups.

Note that CICS performance data contains user identification but for DB2 you must specify either AUTHID=USER for the actual user or AUTHID=GROUP for a RACF group in the entry and pool thread definitions in the resource control table (RCT). Thread reuse increases the overhead for a transaction as it incurs signon processing each time the user changes.

3.2.1.3 Performance

The required performance of an application is primarily an input to the design process. The performance, or service level, is usually defined in terms of the available hours of operation and the time taken to achieve a task for a given transaction throughput, while allowing for exceptional situations. For example, for online transaction processing, the hours of operation may be 6 a.m. until 10 p.m., seven days per week, with an availability target of 99.9% over a six-month period. The time taken to achieve a task may be expressed as a requirement that 95% of all transactions executed during the peak hour should complete within less than 1 sec.

The service level is agreed on by users and the operations or service delivery department and documented in a service level agreement. Even if the level of service is not formally agreed on, there is always an implicit service level agreement, that is, users always expect a better service than what they have now.

From a capacity planning perspective, the useful information is the hours of online operation as that will give you the available remaining hours for batch and housekeeping. We recommend that you plan for and provide sufficient capacity for the peak-hour online transaction throughput. Provided the system is balanced, the response time criteria are a matter of application and database design rather than capacity planning. See *Application Design Guidelines for High Performance* (SG24-2233) for a detailed discussion of tuning techniques.

3.2.1.4 Peak-to-Average Ratio

The hardware capacity provided has to be sufficient for the period of peak utilization rather than the average. The peak period for online is typically 1 hr, but a shorter period may be valid in your environment. An application causes resource utilization at different levels in daily, weekly, quarterly, and annual cycles. There may also be unpredictable peaks, caused by external events such as the weather, that necessitate spare capacity as a safety measure. If the business plan is expressed in terms of sales per quarter, for example, you have

to make assumptions on the working days per quarter and the peak-to-average ratio on a working day.

3.2.2 Design

An application consists of some or all of the following components:

- Online transactions
- Batch programs
- Adhoc queries
- Database housekeeping

The function, design, and development method for each of these components directly affects the capacity required for their operation. With the substantial improvement in power and price performance of computer hardware, new types of applications have been developed that exploit the power available to increase the function and flexibility of the application.

3.2.2.1 Function

The traditional difference among the components is becoming blurred. Online transactions are using additional CPU time to deliver more function to users and to improve flexibility in the face of continuous change. An online transaction may execute a heavy query, for example, when searching a database for the existence of a caller before creating a new client record. Batch programs may be restructured to execute concurrently with the online transactions. This forces frequent commits to reduce the duration of any update locks.

The window for executing housekeeping utility programs has been severely reduced by users demanding an increase in the availability of the online component. These utility programs now have to be run concurrently with other processing, preferably during periods of low online utilization.

3.2.2.2 Development Method

Traditionally applications have been developed with third generation languages such as COBOL. These applications were tuned for good performance as, relative to today, the machine power was much less and the hardware cost much higher. The need for faster and lower-cost application development drove the development of fourth generation languages and program generators. Whether code is generated or interpreted, it tends to have a much higher CPU cost in the application code than traditional methods. In some cases the factor can be as high as a tenfold increase for online transaction processing. This increase in the cost of the application code alters the ratio of application code CPU time to DB2 CPU time quite dramatically.

3.2.2.3 Database Design

The database design directly affects the capacity required. The number of database calls per transaction has steadily increased over time. A number of design techniques contribute to this trend, all of which result in many more DB2 calls than those required to read and update just the business data. Packaged application software solutions use all of the techniques described below to make the package easier to install and tailor for different customers.

Normalization: The output of the logical database design phase in the application development process is a normalized data model. The next phase is the physical database design. The database designer may choose to denormalize

the physical design to improve performance to meet some criteria such as response time. Denormalization compromises the data model — which can make for more complex SQL and reduce the flexibility of the application to handle new business functions. The need to denormalize has decreased with the increase in machine power. The use of normalized models in the physical implementation has also been encouraged by the use of data modeling tools. Using these tools, the developer first creates a fully normalized logical data model. Any modifications to the model are discouraged as they complicate the mapping from logical model to physical implementation. A normalized implementation tends to have more tables, which result in additional CPU time and higher rate of I/O.

Code Tables: Business and validation rules may now be held in tables rather than in the application code to improve the flexibility of the application over time as new functions are required. The data may be simple lookup tables used for validation of and help for data entry fields. In effect, these tables define the domain of an attribute, and some development tools refer to them as *information types*. They are also referred to as *code tables*, as they frequently contain a coded value with a description of the meaning of that code. Some applications can have hundreds if not thousands of these tables. The tables may also represent complex logic, which controls the flow of the application depending on the data content. For example, an insurance company may previously have put the rules for calculating the premium for a policy in the application code. This makes the processing very efficient but means it is more complex to change the rules. Putting the rules in a table improves the flexibility of the application but increases the number of DB2 SQL calls.

History Data: Much more historical data can be kept due to the substantial improvement in the price performance of DASD. For all business transactions, an application may record the previous state of a business object, in the same or another table, the new state, and a record of the transaction that created the change. What used to be a single UPDATE statement is now an UPDATE with two INSERTS.

3.2.3 Technical Environment

To correctly calculate the hardware resource, understand and take into account the technical factors described below.

3.2.3.1 Capture Ratios

The tools that report resource utilization such as DB2 PM and RMF do not account for all of the CPU time used. The reasons for this are discussed in more detail in Chapter 1 of *OS/390 MVS Parallel Sysplex Capacity Planning* (SG24-4680). See Appendix G of the *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957) for a concise summary of monitoring tools and DB2 traces. In this section, we focus on those items in RMF and DB2 PM reports that are relevant to capacity planning and ignore the wealth of other vital data that supports such activities as performance tuning and problem diagnosis. The simplest way to collect and analyze data from all sources is to use IBM Performance Reporter for MVS which has the advantage of keeping the data in DB2 tables.

There are four levels at which resource utilization can be measured.

- Level **1**: OS/390 system utilization

This is reported in the RMF CPU activity report for a specified period of time for a single OS/390 image as a percentage of utilization of all CPUs allocated to that image. The utilization you use for capacity planning depends on whether OS/390 is executing in a logical partition (LPAR) or not.

- OS/390 not executing in an LPAR

  This is the simpler case. If the LPAR BUSY TIME PERC column contains "– – – – " then OS/390 is not executing in an LPAR. In this case, the utilization you use is reported in the MVS BUSY TIME PERC column on the TOTAL/AVERAGE line. It is actually captured as the wait time per CPU. Take note of the number of CPUs as you will need this later to calculate the system and application capture ratios.

- OS/390 executing in an LPAR

  This is slightly more complex. If the LPAR BUSY TIME PERC column contains real values, OS/390 is executing in an LPAR. In this case, the utilization you use is reported in the LPAR BUSY TIME PERC column on the TOTAL/AVERAGE line. To find the number of CPUs, locate the RMF partition data report and use the value in the NUMBER OF LOG PRCRS (number of logical processors) column.

- Level **2**: Application utilization by performance group

  This is reported in the RMF workload activity report in the -SERVICE RATES- column on the APPL% row for each performance group. This is the percentage utilization of a single CPU and could therefore validly exceed 100% if the CPC has more than one CPU. Workloads, such as DB2 or CICS, are allocated to performance groups in the installation performance specification (IPS) data set (SYS1.PARMLIB(IEAIPSxx)). Distinguish carefully between performance group numbers (PGNs) and reporting performance group numbers (RPGNs); otherwise you may mistakenly double-count the CPC utilization. The workload activity report sums this number for all performance groups and reports it for PGN = ALL.

- Level **3**: Transaction task control block (TCB) and service request block (SRB) time for an online transaction or a JES batch job.

  For CICS this is recorded in System Management Facility (SMF) type 110 records and can be reported by using the sample formatting utility DFH$MOLS or equivalent product. The CPU time captured in the SMF type 110 records does not include the DB2 CPU time. To get this data you must turn on the CICS performance trace by setting MNPER=YES in the CICS SYSIN parameters. Be aware that the overhead can be significant when the CICS monitoring facility (CMF) is used to gather performance information. An alternative, with a lower CPU overhead, is to use CICS event monitoring by setting MNEVE=YES. This will tell you the CICS transaction count and elapsed times but not the CPU time by transaction. Define an RPGN for each CICS transaction or group of transactions in the SYS1.PARMLIB(IEAICSxx) data set and use the RMF workload activity report to get a transaction count and average response time. For IMS this is recorded in the IMS log and can be reported on by an IMS utility. For a JES2 job, the elapsed time and CPU time are recorded in SMF type 30 records.

- Level **4**: DB2 TCB CPU time by thread or unit of work

  DB2 statistics, accounting, and performance trace data are recorded in SMF records type 100, 101, and 102 respectively. You can use DB2 PM to produce accounting reports by connection, correlation ID, which for CICS contains the

transaction code, plan, or package. Several units of work may be accumulated into a single accounting record if thread reuse is occurring, depending on the attach control parameters.

The DB2 class 1 accounting CPU time includes the application code CPU time except for CICS. For CICS the difference between class 1 and class 2 CPU time is the time used by the CICS attach code, which is executing under the control of a CICS subtask TCB. As the CPU consumed by the attach code is consistent, there is little added value in using the accounting class 2 trace in a CICS environment unless you suspect a performance problem in the attach code itself. The DB2 accounting class 1 CPU time does not include the CPU time in the CICS application code, which executes under the CICS main-task TCB.

Note that the unit of measurement is different for each level and is further complicated by the fact that many CPCs have more than one engine or CPU. To be able to compare the measurements at the different levels, translate the TCB and SRB CPU times into a percentage utilization of a single CPU.

For capacity planning, the input is often at the lowest level, but you need to calculate the expected utilization at the OS/390 system level, as that is what must be installed. As input to the calculation, we recommend that you use the following capture ratios:

- System capture ratio
- Application capture ratio
- Transaction capture ratio

If you have existing systems, calculate these capture ratios on a regular basis and at a minimum before the start of each capacity planning exercise. Some of the measurements, for example, the CICS performance trace, may have a significant overhead in your environment. We recommend that you measure the overhead and adopt a suitable monitoring strategy to support capacity planning. For example, you could collect the performance data on a nonpeak day just once a week or even once a month. This would allow you to monitor any changes to the capture ratios over time.

Figure 4 on page 49 illustrates three capture ratios (CR) you can refer to while reading the more precise definitions that follow.

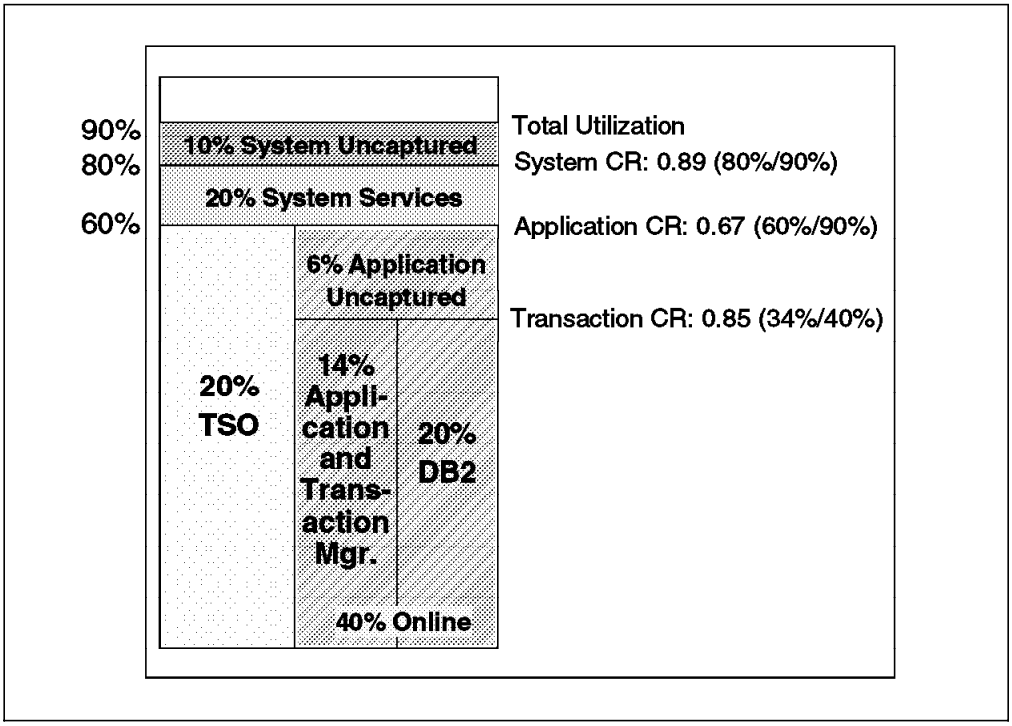


Figure 4. Capture Ratio Example

System Capture Ratio: The system capture ratio is the simplest to calculate and tends to be more stable than the application capture ratio across different installations and workload mixes. Unfortunately it is also the least useful. It is defined as the ratio of the IPS performance group application utilization (see 3.2.3.1, “Capture Ratios” on page 46 level **2**) to the OS/390 system utilization (see 3.2.3.1, “Capture Ratios” on page 46 level **1**). The formula is given in Figure 5.

$$\text{System CR} = \frac{\text{Sum of utilization for ALL IPS performance groups } \mathbf{2}}{\text{N} \times \text{OS/390 system utilization } \mathbf{1}}$$

where N is the number of CPUs available to this OS/390 image.

Figure 5. System Capture Ratio

This value represents the amount of uncaptured CPU time and is typically between 0.88 and 0.92.

Application Capture Ratio: Besides the uncaptured CPU time, as measured by the system capture ratio, there are a number of performance groups, such as those into which JES2, VTAM, and RMF itself have been allocated, whose resource utilization is not attributable to any single application. The simplest way to capacity plan for these system functions, and at the same time allow for the uncaptured CPU time, is to prorate their consumption across all application performance groups by calculating an application capture ratio.

To calculate the application capture ratio, first group the performance groups defined in the IEAIPSxx data set into business applications, for example, TSO and online as illustrated in Figure 4, and system services. Choose the business application groups according to the planned business transaction data that you have available. For example, you may have three CICS regions in different

performance groups, but from a business and capacity planning point of view, they can be considered as one.

The resources consumed by the system services and IRLM address spaces can be handled in one of two ways. You could either treat them as system resources, in which case their cost would be prorated across all applications, whether or not they used DB2, or prorate their CPU cost across only those applications that access DB2. If you are not yet able to calculate this cost for your own site, a safe estimate would be to allow 10% of the application CPU utilization. Note that the cost of the DB2 address spaces is higher in a data sharing environment.

The application capture ratio is defined as the ratio of the sum of the business application groups' utilization (see 3.2.3.1, "Capture Ratios" on page 46 level **2**) to the OS/390 system utilization (see 3.2.3.1, "Capture Ratios" on page 46 level **1**). Note that by excluding the system services performance groups from the summation, the application capture ratio allows for both the uncaptured CPU and prorates the system services at the same time.

The formula for the application capture ratio is given in Figure 6.

$$\text{Application CR} = \frac{\text{Sum of utilization for all business application groups } \mathbf{2}}{N \times \text{OS/390 system utilization } \mathbf{1}}$$

where N is the number of CPUs available to this OS/390 image.

Figure 6. Application Capture Ratio

Table 6 lists some typical application capture ratios that you may find.

Table 6. Typical Application Capture Ratios for Different Workloads

Workload Type	Capture Ratio	Remarks
Short-running batch	0.75–0.85	
Long-running batch	0.80–0.90	
Engineering or scientific batch	0.85–0.95	
Light CICS transactions	0.75–0.85	Lower in the DB2 and multiple region environment (MRO) environment
Heavy CICS transactions	0.80–0.90	Lower in the DB2 and MRO environment
Light IMS transactions	0.70–0.90	Lower in the DB2 and cross-memory environment
Heavy IMS transactions	0.75–0.95	Lower in the DB2 and cross-memory environment
Light TSO	0.60–0.70	
Normal TSO	0.65–0.75	
Heavy TSO	0.75–0.85	

Transaction Capture Ratio: As at the system level, there is also uncaptured CPU at the transaction level. Before you can calculate the transaction capture ratio, first calculate the percentage utilization at the transaction level for all transactions in the business application group. As a reminder, a business

application group is one or more performance groups defined in the IEAICSxx data set of SYS1.PARMLIB. Each performance group may have allocated to it one or more applications. The definitions in the IEAICSxx data set therefore not only control the performance of the OS/390 system but also determine the granularity of the data available for capacity planning. The transaction utilization includes the CPU time in the transaction manager, the application code, and DB2. See 3.2.3.1, “Capture Ratios” on page 46 levels **3** and **4** for guidance on where to find the CPU times.

The formula for transaction utilization is given in Figure 7.

$$\text{Transaction utilization} = \frac{\text{Sum of transaction TCB+SRB time (level 3 and level 4)}}{\text{Measurement interval}} * 100\%$$

where the units of measurement are the same; for example, seconds.

Figure 7. Transaction Utilization

If the OS/390 image has more than one CPU available, the transaction utilization could validly exceed 100%. For CICS the SMF type 110 record excludes the work done in DB2 and therefore you add the CPU time reported by CMF to the DB2 class 1 TCB time.

The formula for the transaction capture ratio is given in Figure 8.

$$\text{Transaction CR} = \frac{\text{Transaction utilization}}{\text{Business application group utilization (level 2)}}$$

where the transaction utilization has been calculated for all transactions in the business application group.

Figure 8. Transaction Capture Ratio

To avoid the effects of long-running tasks not being reported in a short interval, we recommend that you use the CICS and DB2 accounting from a full day′s operation to calculate the capture ratios. For CICS this ratio may be expected to be between 0.8 and 0.9. For IMS it is close to 1.0.

3.2.3.2 Combining CICS and DB2 Performance Data

CICS writes out a performance record at the end of task (EOT). DB2 can write an accounting record at thread termination or for each unit of work (UOW) by specifying TOKENE=YES in the CICS RCT. CICS Transaction Server for OS/390 V1.2 introduces the possibility of a DB2 accounting record at CICS EOT. Therefore, there need not be a one-to-one relationship between CICS performance data and DB2 accounting data.

If the application programs use the CICS XCTL call to transfer control to another program, the CICS transaction code and the correlation token (also known as the CICS LU6.2 token) remain the same. If the program to which control has been transferred also contains SQL code and, assuming that a CICS SYNCHPOINT occurred before the transfer, DB2 creates a new accounting record for the new UOW but with the original CICS transaction ID. If TOKENE=YES, the CICS LU6.2 token is passed to DB2, but again it does not change after a SYNCHPOINT and XCTL. The effect of this is that the CICS CPU is accumulated and reported as

being consumed by the first transaction. A similar effect can occur in the DB2 PM Accounting report if the correlation ID is used.

The transaction code and the correlation token can be updated by using CICS RETURN IMMEDIATE rather than XCTL. This has a slightly higher CPU time cost but in terms of the medium and large transactions described in 3.3.1.2, "Average Transaction Cost" on page 55, the additional overhead is unlikely to be measurable.

One straightforward way to combine the CICS and DB2 transaction level data is to use IBM Performance Reporter for MVS to aggregate the CPU utilization by user group and business function and then combine the results at the aggregated level rather than at the individual transaction level.

3.2.3.3 Machine Power

The sample CPU times given in this book for an SQL statement or a transaction assume you have an IBM 9672-R15, unless stated otherwise. You can calculate the predicted percentage utilization of a 9672-R15 using these CPU times for your application throughput. Note that this could exceed 100%. We recommend that you then use the CP90 M-Value to translate the percentage utilization you have calculated to the machine that will be used to run your workload. See *OS/390 MVS Parallel Sysplex Capacity Planning* (SG24-4680) for a description of CP90 and the detailed definition of the M-value. The M-value is just a way of relating the workload throughput across different machines.

Appendix A, "CP90 M-Values" on page 135 contains the M-values for a range of IBM processors as of August 27, 1997. The M-value for the IBM 9672-R15 is 2710. As an example, suppose you have calculated that the workload will require 120% of a 9672-R15, but in fact you have a 9121-732. Follow the example in Figure 9 to calculate the utilization of your machine.

M-Value of IBM 9672-R15 = 2710
M-Value of IBM 9121-732 = 3661
Utilization of IBM 9672-R15 = 120%

$$\text{Utilization of IBM 9121-732} = 120 \times \frac{2710}{3661} = 89\%$$

Figure 9. Example of Adjustment to a Different Machine

Note that this is the OS/390 system utilization. If you are interested in the response time, you must take into account the number of engines and the engine speed. You can find a detailed discussion of these factors in Chapter 1 of *OS/390 MVS Parallel Sysplex Capacity Planning* (SG24-4680).

3.2.3.4 Machine Utilization

A single application is rarely allocated dedicated resources but shares the available resources with many other applications. As a capacity planner, you have to plan for sufficient capacity to meet the peak load whenever that might occur. You have to estimate the resources required by the new application and map that onto the existing utilization to understand how peak utilization is affected.

3.2.4 Accuracy of Predictions

In the early stages of application development, typically during the feasibility study and early stages of design, you are unlikely to have a great deal of information available. We recommend that, whatever calculations you make, you attempt to approach the problem from at least two different directions, to check the sensibility of the outcome. For example, for the online component calculate the expected peak hour transaction rate from the planned business transactions. If you also know the expected number of users, you can calculate the average think time. If the average think time is less than 10 sec or more than 60 sec, you may need to verify the input data.

The accuracy of the output is dependent on the accuracy of the input data and the technique adopted for translating this into a resource requirement. The techniques described in this chapter have an accuracy of at best −50% to +200%. Bear in mind that the accuracy of the input will probably be of the same order of magnitude.

If possible compare the new application to an existing application, preferably at your installation, in terms of the throughput and design dimensions discussed in 3.2, “Methodology” on page 42. Identify the important differences and use those to make adjustments to the capacity the existing application is using. In many instances this may be the simplest and most reliable method for capacity planning.

3.3 Online Transaction Processing

Online transactions are normally of short duration (less than 0.5 sec). They may access a set of reference tables for validation, help or business logic, and rules in read-only mode. These tables are usually small and tend to be resident in the virtual buffer pool, backed up by a hiperpool, and therefore have minimal I/O. An online transaction would typically also read and possibly update business data tables. These tables are typically much larger and may have a randomized key resulting in synchronous I/O to both the index and the data pages.

3.3.1 Methodology

The essence of this approach is to calculate a peak hour transaction rate and then multiply by an average cost in CPU time per transaction. By taking into account your transaction and system capture ratios, you can then calculate the machine utilization.

Simplified Utilization Formula

Utilization = Peak hour transaction rate × Average transaction cost

3.3.1.1 Peak Hour Transaction Rate

The peak hour could occur at any time of the day or night but traditionally is either mid-morning or mid-afternoon. As well as the peak hour of the day, there is a peak day of the week and a peak week of the month. If the new application has a different peak period profile, this will have to be taken into account.

If you are not given the peak hour transaction rate, you have to calculate it by some means. If you use the planned business transactions, you have to discuss and agree on the following items with the business managers:

- Number of working days in the planning period

  For example, the business plan may be expressed in terms of orders per month. You then need to agree on the working days per month. You could also take into account the variation in the daily workload if it differs much from the average at this stage, by reducing the working days in the planning period.

- Proportion of the daily transactions processed in the peak hour

  The traditional online transaction workload has a peak mid-morning and another peak mid-afternoon. This is because these workloads are primarily driven by users working through a schedule of tasks. With the arrival of new applications where the transactions are driven by the public more directly by such means as the telephone, an automated teller machine (ATM), or the Internet, the traditional patterns will almost certainly change.

  For a traditional application, the following rule-of-thumb can be used:

  Peak Hour Rule-of-Thumb

  20% of the days transactions are processed in the peak hour

  Verify this rule-of-thumb in your own installation.

- Number of application transactions per business transaction

  This may seem easy to estimate but in practice it is not. In most cases it will take many application transactions for a single business transaction. For example, a product order might only require three transactions to enter all the data. If the customer calls back several times with queries and amendments, however, that could drive a substantial number of follow-on application transactions.

  A simple approach is to find the number of application transactions required for the average case and double it. This makes some allowance for the exceptional cases, the use of help, operator error, and the subsequent rework. For example, if it is expected to take 10 transactions without use of help or making errors to enter all the data required to create a record of a new client in an application, double it to 20 for the purposes of capacity planning.

  Business Transaction to Application Transaction Ratio Rule-of-Thumb

  Double the minimum number of application transactions

Table 7 on page 55 illustrates how the sequence of calculations could be made with a spreadsheet.

Table 7. Business Transaction Analysis: Sequence of Calculations			
Business Transaction	**Number per Month**	**Number per Day**	**Number in Peak Hour**
Take new client details	5000	250	50
...	...	...	...
Total	...	...	...

Note: Assumptions are:

- 20 working days per month
- 20% of daily application transactions processed in the peak hour

3.3.1.2 Average Transaction Cost

The next step is to estimate the average transaction cost. Application transactions can be classified into three types:

- Small

  These are typically simple queries accessing a limited number of tables, for example, menu and help screens, or if with business content then a simple query such as on account status or client details. Simple business data updates may be of this type if the data validation is done in the application code and only a limited number of tables are updated. In some applications, such as banking, that require very high throughput rates, the application code and DB2 calls will be fine tuned to achieve a much lower cost than that given here. In this situation full use is made of the performance tuning techniques available such as thread reuse, lock avoidance, minimal signon and authorization checking, dispatching priority control, and minimal use of performance traces.

- Medium

  These are the transactions that are doing the real work of the application. They will be taking in new data input, validating it against the code tables and existing data, and then updating the business data tables. Examples are taking a new product order or adding a new client. They may well have to update the current status of a business object and its associated history data and insert rows in the transaction log. Despite the amount of work, they are still designed to complete in a short period of time because they will hold many update locks, which can cause contention in a high-throughput application.

- Large

  Large transactions in an online environment are not to be encouraged; however, they undoubtedly exist and must be planned for. They may involve complex validation or calculation routines. For example, an insurance company may be rating a complex commercial policy with many different items being insured in many locations. Another large transaction may result when a user is given the option of searching to find a business object whether it exists or not. For example, a telephone caller might only give his or her surname and town, for example, Smith and London. This could invoke a scan and sort of a large number of rows depending on how the table has been indexed and clustered. The difference with this type of transaction is

that it should not hold any DB2 locks and so can be run with a lower priority than any updating transactions.

Table 8 gives you some guidance on the likely CPU and I/O cost of these types of transactions.

Table 8. Sample Online Transactions. CPU time includes transaction manager, DB2, and application costs.

Transaction Type	SQL Calls	Local Response Time (sec)	CPU Range (ms)	Average CPU (ms)	Synchronous I/Os
Small	18	0.1 to 0.3	10 to 30	15	8
Medium	45	0.3 to 1.0	30 to 100	45	20
Large	440	1.0 +	100 +	175	50

The CPU time assumes an operational environment found in customer installations rather than one in which tuning techniques have been used to achieve the best possible throughput rate. The SQL content of these transactions is described in detail in 3.3.2, “Simple SQL Data Manipulation Formulas” on page 58. The DB2 CPU cost is estimated using simple formulas based on the SQL statement type. The resulting CPU time is doubled to allow for the transaction manager and application CPU cost including an allowance for the uncaptured CPU discussed in “Transaction Capture Ratio” on page 50. Allowing for the application to cost as much as DB2 could well be an overestimate depending on the factors discussed in 3.2.2, “Design” on page 45. Calculate the proportion of DB2 to application CPU cost for existing applications in your installation. If you have a new application, you can then use your proportions and adjust according to any known changes in application design for the new application.

You should now estimate the number of each of these transaction types for each of the primary business transactions for which you have calculated a peak hour rate. Table 9 illustrates how this could be done using a spreadsheet.

Table 9. Business Transaction Analysis: Peak Hour Application Transaction Rate Calculation

Business Trans-action	Business Trans-actions in Peak Hour	Application Transactions per Business Transaction			Application Transactions in Peak Hour			
		Small	Medium	Large	Small	Medium	Large	Total
Take new client details	50	3	6	1	150	300	50	500
...	..	..	..	..	..	..	..	..
Totals	2000	-	-	-	40000	45000	5000	90000

You should include in the worksheet in Table 9 an allowance for those business transactions that were not modeled. You can now estimate the number of application transactions to be executed in the peak hour. You should verify this number by comparing it to the expected number of users and if possible other similar existing applications.

The final step is to combine the peak hour transaction rate with the average cost figures. Table 10 illustrates how this could be done using a spreadsheet.

Table 10. Business Transaction Analysis: CPU Utilization and I/O Rate Calculation

Application Transaction Type	Application Transactions in Peak Hour	Trans-action CPU Cost (ms)	Trans-action Number of I/Os	Trans-action rate per second	CPU seconds per elapsed second	I/Os per second
Small	40000	15	8	11.11	0.17	89
Medium	45000	45	20	12.50	0.56	250
Large	5000	175	50	1.39	0.24	69
Total	90000			25.00	0.97	408

In the example in Table 10, the resulting CPU time in seconds per elapsed second is 0.97. This is the same as an application utilization of 97% of an IBM 9672-R15. To make an allowance for uncaptured CPU time at the system level and the system services address space, use the application capture ratio defined in “Application Capture Ratio” on page 49. Follow the example in Figure 10 to calculate the utilization at the system level.

$$\text{System utilization} = \frac{\text{Application utilization}}{\text{Application capture ratio}}\%$$

Example:
Application utilization = 97%
Application capture ratio = 0.8

$$\text{System utilization} = \frac{97}{0.8} = 122\%$$

Figure 10. Example of System Utilization Calculation

If you do not have the detailed business data input, you have to make a judgment as to the nature of the application, taking into account the factors described previously. Table 11 on page 58 illustrates how the sample transactions can be built into three applications, light, medium, and heavy, using a different proportion of each transaction type. Use the average CPU and I/O of the application type that most closely matches the application you are developing.

Table 11. Sample Applications

Application Type	Transaction %			Average Number of SQL Calls	Average CPU Time (ms)	Number of Synchronous I/Os
	Small	Medium	Large			
Light	80	15	5	43	28	12
Medium	15	80	5	61	47	20
Heavy	15	45	15	177	84	28

3.3.2 Simple SQL Data Manipulation Formulas

In this section we present some formulas for estimating the cost of simple SQL statements. By *simple* we mean accessing and updating a single row through an equals unique index access, a matching index scan, or a cursor operation on a single table. In Table 12, for each noncursor SQL statement, we present a lower and upper bound for the cost in CPU time expressed in milliseconds (ms) of an IBM 9672-R15. Select the lower limit for a small table, that is, thousands of rows and up to 10 columns, and the upper limit for a large table, that is, millions of rows and 100 or more columns.

Use DB2 Estimator for greater accuracy or more complex SQL statements.

Table 12. CPU Formulas for Single-Row SQL Calls. CPU times are in milliseconds for IBM 9672-R15.

SQL Statement Type	Lower Bound (ms) [1]	Upper Bound (ms) [2]
Select	0.4	0.8
Open, one fetch, close	0.6	1.0
Each additional fetch	0.1	0.2
Insert, Update, Delete	0.4	0.8
Each index affected [3]	0.2	0.4
Create or terminate thread and commit	2.0	2.0
Transaction manager and application	4.0	50.0

Note:

[1] Lower bound for a small table (two index levels) with few columns (up to 10).

[2] Upper bound for large table (four index levels) with many columns (50 to 100 or more).

[3] Double if update of indexed column

Table 13 on page 59 presents simple formulas for calculating the cost of multiple-row SQL cursor select statements when many rows are fetched in a query type of application. As with single-row statements, use DB2 Estimator for greater accuracy and more complex statements such as those using column functions or subselects.

Table 13. CPU Formulas for Multiple-Row Cursor Select Statements. CPU times are in microseconds for an IBM 9672-R15.

SQL Activity	CPU Time (µs)
Simple formula for many fetches	
Each fetch if many fetches	120
Complex formula for sorting and/or scanned>>fetched	
Each scanned row	4 (1)
Each sorted row	20
Each scanned page	40
Each fetched row	80

Note:

1. Less CPU time (1 µs) if more rows scanned but not returned, with small number of search arguments, because of higher hit ratio in high-speed processor cache

3.3.3 Sample Transaction Detail

In this section we present the detail behind the three sample transactions introduced in 3.3.1.2, “Average Transaction Cost” on page 55. Table 14 on page 60 illustrates the use of the formulas from 3.3.2, “Simple SQL Data Manipulation Formulas” on page 58 applied to the three sample transaction types.

Table 14. Sample Transactions.
CPU time in milliseconds for an IBM 9672-R15.

SQL Statement Type	Small		Medium		Large	
	Number	CPU (ms)	Number	CPU (ms)	Number	CPU (ms)
Create thread and commit	1	2.0	1	2.5	1	3.0
Select - LB [1]	3	1.2	6	2.4	5	2.0
Select - UB [2]	3	2.4	12	9.6	15	12.0
Open, fetch, and close - LB	1	0.6	1	0.6	1	0.6
Additional fetch - LB	8	0.8	10	1.0	160	16.0
Open, fetch, and close - UB	0	0.0	1	1.0	1	1.0
Additional fetch - UB	0	0.0	10	2.0	160	32.0
Update - LB	1	0.5	1	0.5	4	2.0
Update - UB	0	0.0	1	1.0	4	4.0
Insert or delete - LB	0	0.0	1	0.8	6	4.8
Insert or delete - UB	0	0.0	1	1.6	6	9.6
Total SQL calls	17		45		363	
Total DB2 CPU		7.5		23.0		87.0
Transaction CPU [3]		15		46		174
Number of I/Os		8		20		50

Note:

[1] LB - Lower Bound - around 1,000 rows and 10 columns

[2] UB - Upper Bound - around 1 million rows and 100 columns

[3] Assume that the transaction manager and application code CPU time is the same as the DB2 CPU time. This includes the uncaptured CPU time, so there is no need to apply the transaction capture ratio to these numbers. Validate these assumptions for your applications.

Assumptions:

- Average number of indexes per table is 2.
- Indexes touched per UPDATE is 0.5.

If you have detailed information about both the table and index design, the SQL calls, and the frequency of calls, you could use DB2 Estimator for greater accuracy in situations outside the assumptions made for the formulas given in 3.3.2, “Simple SQL Data Manipulation Formulas” on page 58.

3.4 Batch

A traditionally designed batch job accessing the same data as the online component of an application would cause serious contention and is not practicable. For this reason batch programs are executed while the online component is closed down, often known as the *batch window*. For reasons such as global reach across multiple time zones and extended online operation for direct customer service, the batch window is now severely reduced and in many cases nonexistent. This is forcing batch jobs to be redesigned so that they behave in a way compatible with online processing. If this is the case, you can use the same approach as for online transaction processing.

The first step in capacity planning for a true batch environment is to establish those jobs that consume most resources and are on the critical path. The jobs on the critical path determine the elapsed time of the batch schedule and therefore whether or not the schedule can complete within the allotted time span. In the past, the I/O time was the design constraint. This is less true today as there is now an opportunity to exploit the parallelism introduced by DB2 V3 and further extended in V4 and V5. The issue is one of application and database design, as the hardware capacity provided for online is often sufficient for the batch processing. See *Application Design Guidelines for High Performance* (SG24-2233) for a comprehensive discussion of batch program design issues.

For a first estimate, you could use the formulas described in 3.3.2, “Simple SQL Data Manipulation Formulas” on page 58 to estimate the DB2 CPU time of a batch program. Use DB2 Estimator for a more detailed estimate, including the I/O time.

3.5 Query

Estimating the capacity required for a query application is not at all straightforward as, by their very nature, queries are not as well understood in advance as, say, online applications. The business benefits are also harder to quantify, so the capacity may be based on what is affordable rather than a capacity plan built from a business plan.

If you have a known database and SQL query design, you can use DB2 Estimator to estimate the CPU and response time of an individual query. This will not tell you about the performance with many users submitting a mixture of concurrent light, medium, and heavy queries against a database of possibly hundreds of gigabytes. In this instance, you could make use of the IBM Teraplex Centers.

The Teraplex Centers are dynamic laboratories for worldwide usage, testing scalability of high-end platforms, applications, and tools for business intelligence. The centers enable the integration and verification of entire solutions on a much larger scale. Real world stresses such as handling very large number of users, combinations of simple and complex queries, and multiple functions are used to push the solutions to the point of failure. Results are fed back to IBM′s research laboratories and Business Partners, thus improving system performance, enhancing features, and capturing requirements and deficiencies at the source. Many projects are cross-industry, cross-laboratory, multivendor efforts that reflect IBM′s ability to assemble diverse expertise worldwide. Contact your IBM representative if you would like more information or refer to:

- *DB2 for S/390 Terabyte Database: Design and Build Experiences* (SG24-2072) which can be viewed on the Internet (search on “teraplex” from www.ibm.com).
- *Data Warehouse Design Using DB2 for OS/390* (SG24-2249) for a detailed discussion of the design principles and a number of examples.

Having created a number of very large databases (VLDBs) and run query workloads against them, the Teraplex staff deduced a number of high-level rules-of-thumb that can be used in the early stages of planning for a data warehouse on S/390. The information is subject to change and should not be used in place of a detailed evaluation of your actual workload. No explicit claims or guarantees are made as to the accuracy of the information.

In summary, the input to the rules-of-thumb is the quantity of raw data. Using this we calculate the number of engines required to process that quantity of data. Knowing the number of engines we can deduce the I/O bandwidth and from that the number of DASD control units (CUs) needed for a balanced system.

Start by calculating the quantity of raw data in the data warehouse. The raw data is simply the sum of the number of rows in each table multiplied by the row length. Raw data therefore excludes index space, sort space, and system space. The formula for raw data is given in Figure 11.

$$\text{Raw data} = \sum_{i=1}^{n} \text{Row length (bytes)}_i \times \text{Number of rows}_i \text{ (bytes)}$$

where n is the number of tables in the warehouse.

Figure 11. Raw Data Calculation

The next step is to calculate the number of CPUs or engines required to process the raw data. Table 15 gives the lower and upper bound in gigabytes for each generation of IBM CMOS processor.

Table 15. Number of Gigabytes of Raw Data per CPU

CMOS Generation	**Lower Bound (GB)**	**Upper Bound (GB)**
Generation 2 (IBM 9672-R_2 and 9672-R_3)	20	30
Generation 3 (IBM 9672-R_4)	40	60
Generation 4 (IBM 9672-R_5)	100	160

The next step is to calculate the DASD CU I/O bandwidth based on the type of processor. Table 16 on page 63 gives the lower and upper bound in megabytes per second for each generation of IBM CMOS processor.

Table 16. I/O Bandwidth by Processor. Number of megabytes per Second per CPU (engine).

CMOS Generation	Lower Bound MB/sec 1	Upper Bound MB/sec 2
Generation 2 (IBM 9672-R_2 and 9672-R_3)	5	10
Generation 3 (IBM 9672-R_4)	10	20
Generation 4 (IBM 9672-R_5)	13	26

Note:

1 The lower bound applies to a CPU-intensive workload.

2 The upper bound is the maximum scan rate and applies to an I/O intensive workload.

To estimate the DASD requirement, use the formula in Figure 12.

DASD = 3 × Raw data (GB)

This allows for:

- Indexes
- Free space
- Work files
- Active and archive logs
- DB2 directory and catalog
- Temporary tables

Figure 12. DASD Calculation

If you use DB2 compression, you can expect a compression factor of 50%. DB2 compresses table spaces only, not indexes, work files, or logs. Be aware that if you use DB2 compression and the IBM RAMAC Virtual Array (RVA), the RVA compression ratio will be less than the 3.2 to 3.6 compression ratio for uncompressed source data.

To estimate the number of channels, plan for one channel for every 5 MB per second of I/O bandwidth by using the formula in Figure 13.

$$\text{Number of Channels} = \frac{\text{I/O bandwidth (MB per second)}}{5}$$

Figure 13. Channel Calculation

Support of concurrent query users, all exploiting query parallelism, requires substantial memory. Plan for 2 GB of central storage plus 1 GB or more of expanded storage if you exploit hiperpools. As data tends not to be reused with large table scans, large caches in the DASD type may not be needed. You may require DASD cache to support write processing, depending on how you load and update the data in the data warehouse.

The final stage is to estimate the number of DASD controllers required, which is determined by the transfer rates rather than the data capacity. Table 17 on

page 64 gives the VSAM transfer rates driven by DB2 sequential prefetch for a number of IBM DASD controllers.

Table 17. VSAM Transfer Rates for IBM Controllers

IBM DASD Type	Transfer Rate MB/sec
RAMAC I	11.5
RAMAC II	11.5
RAMAC III (4-path single controller)	22.3
RAMAC III (9390-2, 8 path dual controller)	44.6
RAMAC Virtual Array 2 Model T42 (4-path)	22.0
RAMAC Virtual Array 2 Model T82 (8-path)	27.0
RAMAC Scalable Array (16-path)	49.0
RAMAC Scalable Array 2 (6 Data Controller pairs)	55.0

As an example, assume:

- 400 GB of raw data
- IBM CMOS generation 4 processors
- IBM RAMAC Virtual Array model T82
- Workload mix of both I/O and CPU bound queries implying midpoint of lower and upper bound for I/O bandwidth calculation

Refer to Figure 14 for the sequence of calculations.

Processor calculation

Number of CPUs = $\frac{400}{160} = 3$ (rounded up)

The stand-alone processor would be an IBM 9672-R55.

I/O bandwidth calculation

I/O bandwidth = $3 \times 26 = 78$ (MB per second)

Channel calculation

Minimum number of channels = $\frac{78}{3} = 26$

DASD controllers calculation

RVA2 controllers = $\frac{78}{27} = 3$ (rounded up)

Assuming each controller has eight channels as recommended, this is almost equal to the minimum calculated above.

DASD capacity calculation

DASD = $3 \times 400 = 1200$ (gigabytes)

Given the four controllers, implies 300 GB per controller

Central and expanded storage

For a high number of concurrent queries, we would recommend a minimum of 2 GB of central storage plus 1 to 2 GB of expanded storage.

Figure 14. Sample Capacity Calculation for a Query Workload

3.6 Buffer Pool Sizing

DB2 is designed to exploit large buffer pools and thereby avoid I/Os, the response time of which, relative to the speed of processors, has hardly changed over many years. The simplest method to estimate the buffer pool size is by relation to the power of the processor. If you have no other data available, use the formula in Figure 15. This formula is empirical, that is, it is based on experience and, if possible, should be verified in your own installation. We are not concerned here with how many buffer pools to have nor how page sets or partitions should be allocated to them; that is a tuning consideration.

$$\text{Buffer pool} = 40 + \frac{\text{Processor M-value}}{10} \text{ (megabytes)}$$

Figure 15. Buffer Pool Size Based on Processor

The initial constant of 40 MB is to allow for application reference data and a proportion of the index nonleaf pages. For example, this formula would result in a total buffer pool size of 311 MB for an IBM 9672-R15. See Appendix A, “CP90 M-Values” on page 135 for a list of IBM processors and their M-values.

An even simpler formula for an online application is to use the expected transaction rate as shown in Figure 16.

$$\text{Buffer pool} = 40 + \text{Transactions per second} \times 10 \text{ (megabytes)}$$

Figure 16. Buffer Pool Size Based on Transaction Rate

If you have an existing system, adjust the buffer pool size while observing the effect on paging to expanded storage and DASD, the DB2 I/O rate, and the buffer pool hit rate. Unfortunately it is not easy to understand at what point to stop adjusting these measures as there is no such thing as the ideal buffer pool hit rate.

A more rigorous approach is to increase the buffer pools until the number of rereads in a fixed period of time, such as 5 min, is close to zero. The simple definition of a reread is when a page is read more than once in a predetermined period of time. To calculate the number of rereads, you need to use performance trace class 4 and IFCID 6. Use the online monitor of DB2 Performance Monitor to initiate the trace and capture the data for a fixed period of time. Start the performance trace for class 4 and IFCID 6 only, otherwise you could cause a serious performance degradation if you started all performance trace classes. You can then use the batch reporting component to format the records into a printed report. For an example of the SQL and LOAD utility control statements you could use, see Appendix C, “Buffer Pool Tuning” on page 143.

3.7 DASD Space

The simple estimate for DASD space is based on the calculation of the amount of raw data. The raw data is simply the sum of the number of rows in each table multiplied by the row length. Raw data therefore excludes index space, sort space, and system space. If you do not yet know the size and specification of all the tables, identify the largest tables only and apply the 80:20 rule, that is, 20%

of the tables account for 80% of the space, by adding 20% to the raw data size of the largest tables. The formula for application raw data is given in Figure 17 on page 66.

$$\text{Application raw data} = 1.2 \times \sum_{i=1}^{n} \text{Row length (bytes)}_i \times \text{Number of rows}_i \text{ (bytes)}$$

where n is the number of large tables.

Figure 17. Application Raw Data Calculation

Multiply the raw data by 2 to get an estimate of the space needed by the application data, that is, including an allowance for indexes and free space within a DB2 page set or partition. To allow for the DB2 logs, archive logs, work files, directory, and catalog, multiply the raw data amount by 2.5.

3.7.1 Compression

Using the COMPRESS clause of the CREATE TABLESPACE and ALTER TABLESPACE SQL statements allows you to compress data in a table space or in a partition of a partitioned table space. In many cases, using the COMPRESS clause can significantly reduce the amount of DASD space needed to store data. You may expect to achieve compression ratios between 2:1 to 8:1 depending on the characteristics of your data. In addition, with compressed data, the amount of I/O required to scan a table space can be reduced.

3.7.1.1 Performance Considerations

To estimate how well a given table space will compress, run the stand-alone utility, DSN1COMP. Consider the following when making your decision:

- Compressing data can result in a higher processor cost, depending on the SQL workload. However, if you use IBM′s synchronous data compression hardware, processor time is significantly less than if you use the DB2-provided software simulation or an edit or field procedure to compress the data. Indexes are not compressed.
- The processor cost to decode a row with the COMPRESS clause is significantly less than the cost to encode that same row. This rule applies regardless of whether the compression uses the synchronous data compression hardware or the software simulation built into DB2.
- When rows are accessed sequentially, fewer I/Os might be required to access data stored in a compressed table space. However, there is a trade-off between reduced I/O resource consumption and the extra processor cost for decoding the data.
- If random I/O is necessary to access the data, the number of I/Os will not decrease significantly, unless the associated buffer pool is larger than the table and the other applications require little concurrent buffer pool usage.
- Depending on the SQL workload, as the degree of data compression increases, it can lead to:
  - Higher buffer pool hit ratios
  - Fewer I/Os
  - Fewer getpage operations

- The data access path DB2 uses affects the processor cost for data compression. In general, the relative overhead of compression is higher for table space scans and less costly for index access.
- Some types of data compress better than others. Data that contains hexadecimal characters or strings that occur with high frequency compresses quite well, whereas data that contains random byte frequencies might not compress at all. For example, textual and decimal data tends to compress well because of the high frequency of certain byte strings.
- In determining whether using the COMPRESS clause is advantageous, examine the uncompressed length of a row. As row lengths become shorter, compression yields diminishing returns because 8 bytes of overhead are required to store each record in a data page. However, when row lengths are very long, compression of the data portion of the row may yield little or no reduction in data set size because DB2 rows cannot span data pages. In the case of very long rows, using a 32 KB page can enhance the benefits of compression, especially if the data is accessed primarily in a sequential mode.

A number of installations have found that the CPU costs roughly balance out, with the primary benefit being the reduced buffer pool size, fewer I/Os, and less DASD for a given application workload. If you are constrained on buffer pool size, I/O bandwidth, or DASD capacity, it may be worth investigating compression for your applications.

3.7.1.2 Tuning Recommendations

There are some cases where using compressed data results in an increase in the number of getpages, lock requests, and synchronous read I/Os. There are two possible reasons for this:

- Sometimes updated compressed rows cannot fit in the home page, and they must be stored in the overflow page. This can cause additional getpage and lock requests. If a page contains compressed fixed-length rows with no free space, there is a great probability that an updated row has to be stored in the overflow page.

  To avoid the potential problem of more getpage and lock requests, add more free space within the page. Start with 10% additional free space and adjust further as needed. If, for example, 10% free space was used without compression, then start with 20% free space with compression for most cases. This recommendation is especially important for data that is heavily updated.
- If your lock request count increases, but the number of getpages does not increase, determine whether lock escalation has occurred. When data is compressed, more rows are stored in a page and thus more rows are processed before lock escalation occurs. Without compression, lock escalation takes place earlier, and so no more lock requests are issued at an earlier time during query processing.

3.7.2 Work Files

DB2 work files are used whenever temporary space is required to support SQL processing, for example:

- Sorts
- Joins

- View materialization
- Temporary tables

Work files are shared by all users of a DB2 subsystem, so the space allocated must be sufficient for concurrent processing. See Chapter 5 in the *DB2 for OS/390 Version 5 Administration Guide* (SC26-8957) under “Create Additional Work File Table Spaces” and “Controlling Sort Pool Size and Sort Processing” for recommendations on the number of work files and a description of how to calculate the space required to support a sort.

The sort begins with the input phase when ordered sets of rows are written to work files. At the end of the input phase, when all the rows have been sorted and inserted into the work files, the work files are merged together, if necessary, into one work file containing the sorted data. The merge phase is skipped if there is only one work file at the end of the input phase. In some cases, intermediate merging might be needed if the maximum number of sort work files has been allocated. The work files used in sort are logical work files, which reside in work file table spaces in your work file database (which is DSNDB07 in a non-data-sharing environment). DB2 uses the buffer pool when writing to the logical work file, so, depending on the size of the sort, there may be no I/O at all. The number of work files that can be used for sorting is limited only by the buffer pool size when you have the sort assist hardware.

If you do not have the sort hardware, up to 140 logical work files can be allocated per sort, and up to 255 work files can be allocated per user.

It is possible for a sort to complete in the buffer pool without I/Os. This is the ideal situation, but it might be unlikely, especially if the amount of data being sorted is large. The sort row size is actually made up of the columns being sorted (the sort key length) and the columns the user selects (the sort data length).

When your application needs to sort data, the work files are allocated on a least recently used basis for a particular sort. For example, if five logical work files (LWFs) are to be used in the sort, and the installation has three work file table spaces (WFTSs) allocated, then:

- LWF 1 would be on WFTS 1.
- LWF 2 would be on WFTS 2.
- LWF 3 would be on WFTS 3.
- LWF 4 would be on WFTS 1.
- LWF 5 would be on WFTS 2.

To support large sorts, DB2 can allocate a single, logical work file to several physical work file table spaces.

If the data input to the sort is already close to the sort sequence, DB2 continues to use the first LWF. If this LWF is allocated to a work file table space with secondary extents, the table space is extended by the full 119 secondary extents before another work file table space is used. Due to the cost of VSAM allocation and DB2 formatting, we recommend that you do not define secondary extents for work file table spaces with the exception of one, preferably the last defined table space. Use this table space as a safety valve for the capacity of all work files by monitoring the number of secondary extents. If this one table space goes into secondary extents, it may mean that you need more capacity to support the work

files. Either enlarge the existing work file table spaces or, if the I/O rate to the existing work file DASD volumes is causing contention and poor response times, define an additional work file table space on a new volume.

Chapter 4. Capacity Planning for DB2 Utilities

In this chapter we provide simple formulas and lookup tables for some utilities and explain them. We discuss capacity planning for the LOAD, RECOVER, and COPY utilities.

4.1 Sample Table Definitions

In this section we present the table definitions used throughout this chapter.

To test our formulas we created a set of tables in DB2 Estimator. The 64 GB table has 64 partitions. All other tables have eight partitions.

Each table has two nonpartitioning indexes and one partitioning index. Each index has three columns and is unique.

The sizes were chosen to demonstrate utility times when multiplying table sizes by 100. As the maximum size is 64 GB, we started out with 6.4 MB, that is, 6,400 KB. With the sizes chosen we cover the full range of time a LOAD utility can use.

We chose a typical number of columns: a minimum of 20 and a maximum of 100.

For column type we chose CHAR, as the LOAD performance with CHAR columns is comparable to that with INTEGER columns. Assuming that the average column size is constant in all tables, we had the column size defined by the table with the maximum number of 100 columns. Therefore, the maximum column size was defined by the maximum page size (between 4046 and 4056) divided by the number of columns (100), that is, 4056 / 100 = 40.

Table 18 gives the number of rows we can load, given the table size and the specified number of columns. The theoretical maximum includes the header page and the space map pages. However, the theoretical maximum has not been used.

This set of tables uses as a comparison basis a constant number of bytes per column and a constant number of bytes per table space or partition, varying the number of rows and the number of columns.

Table 18. Number of Rows (Column Type Is CHAR(5))

Table Size	**Number of Rows**	
	20 Columns	**100 Columns**
6,400 KB	62,072	13,420
640 MB	6,206,907	1,342,032
64 GB	620,690,688	134,203,200

Using the sizes in Table 18, we find the size in kilobytes as listed in Table 19, using DB2 Estimator.

Table 19. Data Size in Kilobytes		
Table Size	**Data Size in Kilobytes**	
	20 Columns	**100 Columns**
6,400 KB	6,720	6,720
640 MB	671,088	671,484
64 GB	67,108,864	67,108,864

Table 20 lists the number of pages in our tables, as calculated by DB2 Estimator.

Table 20. Number of Pages		
Table Size	**Number of Pages**	
	20 Columns	**100 Columns**
6,400 KB	1,680	1,680
640 MB	167,772	167,772
64 GB	16,777,088	16,777,088

In 4.4, “Parallel LOAD of 64 GB Table” on page 79, we use another set of tables with a constant number of bytes (64 GB) and a constant number of columns (100), varying the length of the columns and the number of rows. The number of rows, given the number of characters per column used in that set of tables, is shown in Table 21.

Table 21. Number of Rows (Each Row Has 100 Columns)			
Table Size	**Number of Rows**		
	CHAR(5)	**CHAR(10)**	**CHAR(20)**
64 GB	134,203,392	67,101,696	33,550,848

4.2 LOAD Table

We look at the LOAD utility performance in more detail.

4.2.1 Formulas for Elapsed and CPU Times

We show the formulas and the assumptions behind these formulas in Figure 18 on page 73.

$$\text{DB2 V3 CPU time} = NR \times (94 + (21 + 70 \times NPI) + 6.5 \times NC) + 300 \times NP$$

$$\text{DB2 V3 Elapsed time} = \frac{\text{CPU time for DB2 V3}}{90\%}$$

$$\text{DB2 V4 CPU time} = NR \times (94 + (21 + 70 \times NPI) + 1.3 \times NC) + 300 \times NP$$

$$\text{DB2 V4 Elapsed time} = \frac{\text{CPU time for DB2 V4}}{90\%}$$

$$\text{DB2 V5 CPU time} = NR \times (61 + (21 + 62 \times NPI) + 1.2 \times NC) + 300 \times NP$$

$$\text{DB2 V5 Elapsed time} = \frac{\text{CPU time for DB2 V5}}{90\%} \text{ if NPI=0, else}$$

$$\text{DB2 V5 Elapsed time} = \frac{\text{CPU time for DB2 V5}}{110\%}$$

NPI For a partitioned table space, number of nonpartitioning indexes
NPI For a nonpartitioned table space, number of nonclustering indexes
NP Number of pages
NR Number of rows
NC Number of columns

Column processing cost is heavily dependent on column type. "1.2" (µs) shown for DB2 V5 assumes an average CPU time with a predominant use of character and internal integer and decimal. For more accurate information, use the following:

- 1 µs for CHAR or internal integer
- 1.5 µs for internal decimal
- 4 µs for external integer or decimal

Assumptions:

- CPU time in microseconds (µs) on IBM 9672 R63 (single CPU)
- Elapsed time assuming no I/O bottleneck, for row size of a few hundred bytes or less
- SORTKEYS option used

Figure 18. Formulas for LOAD CPU and Elapsed Times

4.2.1.1 Replacing CPU Used with a Different Processor Model

To compute the elapsed times for a different CPU, use the formulas given in Figure 19.

- if faster CPU
  - $\text{WORST} = \frac{\text{Base CPU}}{\text{CPU\%}}$
  - $\text{BEST} = \frac{\text{Base CPU}}{\text{CPU\%}} - (\text{Base CPU} - \text{New CPU})$
- if slower CPU,
  - $\text{WORST} = \frac{\text{Base CPU}}{\text{CPU\%}} + (\text{New CPU} - \text{Base CPU})$
  - $\text{BEST} = \text{Max}(\frac{\text{Base CPU}}{\text{CPU\%}}, \frac{\text{New CPU}}{1.2})$
- $\text{Elapsed time} = \frac{(\text{WORST} + \text{BEST})}{2}$

Figure 19. Formulas to Replace CPU Used with Different Model

Example: Here we present the application for the case of the faster CPU (Figure 20). We replace the IBM 9672 R63 (single CPU) with an IBM 9672 R65. For the M-values used, see Appendix A, “CP90 M-Values” on page 135.

LSPR 9672-R63	5153
LSPR 9672-R65	13001
R63 / R65	$\frac{5153}{13001} = 40\%$
Computed Time	40 minutes @ 90% CPU Utilization
WORST	$\frac{40}{90\%} = 44.4$
BEST	$\frac{40}{90\%} - (40 - 40 \times 40\%) = 20.4$
Elapsed Time	$\frac{(44.4 + 20.4)}{2} = 32.4$

Figure 20. Example to Replace IBM 9672 R63 with IBM 9672 R65

In this formula the fivefold CPU time reduction for column processing is factored in for DB2 V4. For DB2 V5 we have a 35% CPU time reduction for row processing and up to a 40% reduction of elapsed time. The reduction comes from multitasking and overlapped processing for DB2′s internal sort and from the use of virtual storage instead of SYSUT1 and SORTOUT data sets, which eliminates the BSAM I/O times.

In Figure 21 we present the same formula with the difference that you have to apply for DB2 V5 with nonpartitioning indexes.

LSPR 9672-R63	5153
LSPR 9672-R65	13001
R63 / R65	$\frac{5153}{13001} = 40\%$
Computed Time	40 minutes @ 90% CPU Utilization
WORST	$\frac{40}{110\%} = 36.4$
BEST	$\frac{40}{110\%} - (40 - 40 \times 40\%) = 12.4$
Elapsed Time	$\frac{(36.4 + 12.4)}{2} = 24.4$

Figure 21. Example to Replace IBM 9672 R63 with IBM 9672 R65, DB2 V5, NPI

4.2.1.2 Calculated Elapsed Times

Using the formula in Figure 18 on page 73, we calculate the elapsed times at 90% CPU utilization for our nine tables for the three DB2 releases. See Table 22 for the calculated times of DB2 V3. Look at the time for 64 GB and 20 columns (75:18:30 hours) and compare it with the time for 64 GB and 100 columns (39:02:21). The table sizes are the same in both cases. However, the number of rows are different. This is because the number of rows that can be stored in the 64 GB table depends on the number of columns, given that each column is CHAR(5). Processing time is greatly influenced by the number of rows and columns, and less by DASD space used.

Table 22. LOAD Table Elapsed Times for DB2 V3

Table Size	20 Columns	100 Columns
6400 KB	00:00:27	00:00:14
640 MB	00:45:11	00:23:25
64 GB	75:18:30	39:02:21

Formula used:

DB2 V3 = NR * (94 + (21 + 70 * NPI) + 6.5 * NC) + 300 * NP
@ 90% CPU utilization

In DB2 V4 there is a five-fold CPU time reduction for column processing, resulting in overall elapsed time reduction as can be seen in Table 23 on page 76.

Table 23. LOAD Table Elapsed Times for DB2 V4		
Table Size	**20 Columns**	**100 Columns**
6400 KB	00:00:20	00:00:06
640 MB	00:33:14	00:10:30
64 GB	55:23:06	17:30:02
Elapsed Time Reduction Compared to DB2 V3 (%)	26	55
Formula used: DB2 V4 = NR * (94 + (21 + 70 * NPI) + 1.3 * NC) + 300 * NP @ 90% CPU utilization.		

If you compare the elapsed times for the 64 GB table, they vary widely depending on the number of columns used (between 55 min and 17 min). It is therefore of no use to know how many gigabytes a table has to predict load times. The next conclusion is that for many rows (the 20 column table has 4.6 times more rows than the 100 column table) CPU time becomes the dominant factor.

In DB2 V5 there is a CPU time reduction for row processing, resulting in a further overall elapsed time reduction as can be seen in Table 24.

Table 24. LOAD Table Elapsed Times for DB2 V5		
Table Size	**20 Columns**	**100 Columns**
6400 KB	00:00:13	00:00:04
640 MB	00:22:24	00:07:24
64 GB	37:19:16	12:19:09
Elapsed Time Reduction Compared to DB2 V4 (%)	23	23
Formula used: DB2 V5 = NR * (61 + (21 + 62 * NPI) + 1.2 * NC) + 300 * NP @ 90% CPU utilization, if NPI=0, else 110%		

4.2.2 Using DB2 Estimator for Elapsed, CPU, and I/O Times

Now that we have presented the theoretical basis for understanding load times in DB2, we present the load times as computed by DB2 Estimator, where the formula is more refined. DB2 Estimator is used for a more detailed analysis. DB2 Estimator is Version 5.0.31, dated 9/22/97. The DASD type used is RAMAC-3 Array Storage, and the CPU model used is the IBM 9672 R63.

Using the same tables shown in Table 18 on page 71 as input, we used DB2 Estimator to calculate the load CPU times. In MVS.CFG, the configuration selection file of DB2 Estimator, we specified 9672-R63 for CPU and RAMAC-3 for

DASD. The elapsed times in Table 25 are comparable to the calculated times in Table 24. However, there are faster methods than just loading a 64 GB table in a single step, as we explain in 4.4, “Parallel LOAD of 64 GB Table” on page 79.

First let us look at Table 25, which represents the elapsed times as found by DB2 Estimator.

Table 25. LOAD Table Elapsed Times for DB2 V5 by DB2 Estimator

Table Size	20 Columns	100 Columns
6400 KB	00:00:19	00:00:05
640 MB	00:26:48	00:07:49
64 GB	44:23:21	12:58:57

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ...

Now let us look at Table 26 for the CPU time. We find again what we have already seen in the formula, that the number of rows influences the load time more than the number of bytes loaded. Here, when the number of rows is 4.6 times higher (620,690,688 / 134,203,200), the CPU time required for the same number of bytes to be loaded is 3.2-fold (36:51:56 / 11:21:21).

Table 26. LOAD Table CPU Times for DB2 V5 by DB2 Estimator

Table Size	20 Columns	100 Columns
6400 KB	00:00:16	00:00:04
640 MB	00:22:08	00:06:49
64 GB	36:51:56	11:21:21

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ...

In Table 27 we present the accumulated I/O times for a complete load, including all sort I/O. That number is an indicator of the I/O bandwidth you have to plan to achieve optimum concurrency. Please keep in mind that an essential part of it is SORTWKnn time. To study this effect in more detail, see 4.4, “Parallel LOAD of 64 GB Table” on page 79. Please bear in mind that we used RAMAC-3 Array Storage devices. Slower devices can double the time easily.

Table 27. LOAD Table I/O Times for DB2 V5 by DB2 Estimator		
Table Size	**20 Columns**	**100 Columns**
6400 KB	00:00:12	00:00:06
640 MB	00:30:32	00:11:05
64 GB	55:43:34	21:23:25
Note: This is equivalent to using the following control statement for the LOAD utility: LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ...		

In conclusion, the impact that a large number of rows and sort have on load times is by far more important than the number of bytes loaded.

4.2.3 Elapsed Time Comparison Using Formula and DB2 Estimator

In this section we apply the formula for faster CPU to DB2 V5 LOAD table elapsed times.

Comparing the times calculated using the formula (Table 28) with the times computed by DB2 Estimator (Table 29), we find that they are sufficiently identical. (Please remember that the formulas are quite simplified.)

Table 28. LOAD Table Elapsed Times for DB2 V5 on IBM 9672 R65		
Table Size	**20 Columns**	**100 Columns**
6400 KB	00:00:09	00:00:03
640 MB	00:15:00	00:04:57
64 GB	25:19:00	08:15:14
Formula used: DB2 V5 = NR * (61 + (21 + 62 * NPI) + 1.2 * NC) + 300 * NP @ 90% CPU utilization, if NPI=0, else 110% and recomputed using Figure 21 on page 75 with 110% for NPI=2		

Table 29. LOAD Table Elapsed Times for DB2 V5 by DB2 Estimator for IBM 9672 R65		
Table Size	**20 Columns**	**100 Columns**
6400 KB	00:00:10	00:00:04
640 MB	00:14:09	00:05:57
64 GB	23:20:38	09:52:01
Note: This is equivalent to using the following control statement for the LOAD utility: LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ... An IBM 9672 R65 is used instead of the IBM 9672 R63 in Table 25 on page 77.		

4.3 LOAD Table Partition

Large tables are designed to be loaded in partitions. In Table 30 we present the load times for a single 1 GB partition to help you have a first estimate of possible loading times. The two nonpartitioning indexes are still present, as is the partitioning index. Therefore, sort times are included in these times, as they are in a normal reload situation. And, as is to be expected, we see a strong dependency of the load times on the number of rows rather than the size.

Table 30. LOAD Table Partition Times for DB2 V5 by DB2 Estimator

Resource	20 Columns	100 Columns
CPU Time	00:58:13	00:13:57
I/O Time	29:32:19	06:28:13
Elapsed Time	10:28:42	02:15:56

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ... PART nnn

4.4 Parallel LOAD of 64 GB Table

Now that we have seen that load times depend more on the number of rows than on the number of bytes, we can study the behavior of large tables with a constant number of columns. In this section we study more closely the relationship between size and load times. To keep the variables to a minimum, we have one parameter—the number of rows—that can change. We vary the number of rows by changing the column length from 5 characters to 10 characters and from 10 characters to 20 characters, thus reducing the number of rows by one-half at each step. The number of columns and the number of bytes in the table space remain constant. Therefore the I/O times for reading and writing the table space remain constant, as long as there is no additional I/O, such as sort. The same logic applies to CPU time and elapsed time. Any difference can be attributed to sort processes for the partitioning index and CPU time for row processing.

For small tables sort does not really matter in most cases. For large tables it has a strong impact, so you should try to lessen that impact by removing the nonpartitioning indexes from the tables to exclude the sort from load times. Chapter 4.5, “RECOVER Index” on page 82 deals with the time required for index recovery. In Table 31 we present the times for a load in a single step to illustrate the difference in sorting time. The table demonstrates the influence that row processing has on CPU time.

Table 31. LOAD Table Times for DB2 V5 by DB2 Estimator (64 GB and 100 Columns)

Resource	Number of Rows			Ratio	
	1	2	3	1 : 2	2 : 3
CPU time	06:33:06	03:57:06	02:40:41	1.66:1	1.48:1
I/O time	17:14:07	14:34:46	14:20:01	1.18:1	1.02:1
Elapsed time	08:00:44 E	07:12:36	07:03:34	1.11:1	1.02:1
Number of rows	134,203,392 1	67,101,696 2	33,550,848 3	2:1	2:1

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ...

The table has no nonpartitioning index.

The Ratio 1 : 2 column represents the relationship between the Number of Rows 1 and Number of Rows 2 columns. The Ratio 2 : 3 column represents the relationship between the Number of Rows 2 and Number of Rows 3 columns. You can see that, for an increasing number of rows, the impact is more on CPU time and less on I/O time and elapsed time.

Looking at the load times for the single partition with no nonpartitioning index in Table 31 with Number of Rows 1, you see that there is not much variation between the CPU time and elapsed time. Thus for this type of table, on a model IBM 9672 R63, a single engine runs at almost 100%, and therefore the number of parallel load jobs must not exceed the number of CPUs available. That becomes better with reduced number of rows.

To exploit parallelism we split the single load job into multiple partition load jobs. Table 32 shows the load times for a single 1 GB partition with only the partitioning index and no nonpartitioning index.

Table 32. LOAD Table Partition Times for DB2 V5 by DB2 Estimator (No Nonpartitioning Index) (tablen)

Resource	Number of Rows		
	1	2	3
CPU time	00:06:09	00:03:42	00:02:31
I/O time	00:14:27	00:12:39	00:12:30
Elapsed time	00:07:31 E	00:06:46	00:06:37
Number of rows	2,096,928 1	1,048,464 2	524,232 3

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ... PART nnn

The table partition size is 1 GB and a row has 100 columns.

The advantage of parallel load depends on the number of CPUs and the access path. If I/O bandwidth and the number of CPUs available allow, the load of 64 partitions (1 GB each) can be organized, for example, in a set of six batch jobs that run concurrently. In this way you would need almost 11 such sets, which could theoretically complete in 00:07:31 × 11 = 01:22:41 (E in Table 32) compared with 08:00:44 (E in Table 31 on page 80).

The same technique applies for 2 and 3 in Table 32. In fact the difference between CPU time and elapsed time allows even a higher degree of parallelism.

To establish the value of this method we add the time required to load a single partition with two nonpartitioning indexes (Table 33 on page 82). Looking at the elapsed time and comparing the I/O bandwidth with the table with no nonpartitioning index, it is obvious that a single partition load done in parallel for all partitions is not very effective when nonpartitioning indexes are present.

Table 33. LOAD Table Partition Times for DB2 V5 by DB2 Estimator (Two Nonpartitioning Indexes) (tableb)

Resource	Number of Rows		
	1	2	3
CPU time	00:32:30	00:20:05	00:13:52
I/O time	06:28:13	05:27:50	04:58:05
Elapsed time	02:15:56	01:54:26	01:43:50
Number of rows	2,096,928 1	1,048,464 2	524,232 3

Note:

This is equivalent to using the following control statement for the LOAD utility:

LOAD DATA RESUME NO REPLACE LOG NO SORTKEYS INTO TABLE ... PART nnn

The table partition size is 1 GB and a row has 100 columns.

Next we look at the time it takes to recover an index after loading the partitions without nonpartitioning indexes.

4.5 RECOVER Index

For large tables it is advisable to do a CREATE INDEX ... DEFER YES. In this way the DB2 sort can be avoided because an external sort is used by the RECOVER utility. Table 34 indicates the time needed to do an index recovery for the nonpartitioning indexes we dropped in 4.4, “Parallel LOAD of 64 GB Table” on page 79 to have a faster partition load time.

Table 34. RECOVER Index Times for DB2 V5 by DB2 Estimator (64 GB and 100 Columns)

Resource	Number of Rows		
	1	2	3
CPU time	02:10:15	01:22:38	00:58:50
Elapsed time	03:50:08	02:26:00	01:43:56
Number of rows	134,203,392 1	67,101,696 2	33,550,848 3

Note:

This is equivalent to using the following control statement for the RECOVER utility:

RECOVER INDEX (...)

Recovering the indexes in parallel would be beneficial too as you could exploit buffering and caching.

Loading the partitions in parallel with the nonpartitioning indexes creates contention on the nonpartitioning indexes that will delay the loading.

4.6 RECOVER Table Space or Partition

In this section we deal with recovery of a single table space or partition to give you an idea of what to look for when recovering. The objective is to be able to recover in a predetermined time for a selected set of table spaces. This time could be part of a service level agreement that you have to fulfill. To help you understand the performance-sensitive variables, we present a formula to calculate elapsed times for recovery (Figure 22).

```
DB2 V3 = 2 ms * number of pages        /* Restore from full image copy */
     + number of log data bytes to be read / 2 MB/sec    /* Log read */
     + 20 ms * number of pages read for update          /* Log apply */
```

- DB2 V3 used as basis.
- CPU used is 3090-180J.

Figure 22. Formula for RECOVER Table Space or Partition Elapsed Time

In Figure 22, we see three distinct parts for recovery time. The first two are independent of the number of updates. These update-independent times refer to the size of the image copy and the size of the log file to be searched (see Table 35). Only the third part depends on the number of updates to the table space or partition. The update times are presented in Table 36.

Table 35. Time to Read Image Copy and Log Files

Image Copy Size (GB)	Log File Size 0 GB	Log File Size 1 GB	Log File Size 4 GB
0	00:00:00	00:08:44	00:34:57
0.5	00:04:16	00:13:00	00:39:13
1.0	00:08:32	00:17:16	00:43:29
2.0	00:17:04	00:25:48	00:52:01
4.0	00:34:08	00:42:52	01:09:05

Note: Apply image copy and log file read times given in the formula in Figure 22.

Now let us assume that you have a service level agreement to recover each table space in a list in at most 30 min.

What conclusions can you derive from Table 35?

- The table space image copy must be smaller than 4 GB.
- In addition, the log file you have to search must be smaller than 4 GB.

What can you do to achieve those goals?

- You have influence on the table space only if it is partitioned, has less than 64 partitions, and you are allowed to change the partitioning.
- Usually you have more influence on the size of the log file to search. Considering that the table space in question is not related to any other table

space by referential integrity or by simultaneous update from applications, you have an option to limit the length of the log file to be searched, that is, to run an image copy or an incremental image copy.
- The other option is to have the table space regularly closed or pseudo-closed (not an option that is easy to maintain).

Comparing the time to read 4 GB of a log file and 4 GB of an image copy file (01:09:05) with the time to read only 4 GB of an image copy file (00:34:08), we find that log file and image copy file are read roughly at the same speed.

Therefore the size of all log files and all image copy files read must not exceed about 3 GB (almost 24 min to read) if recovery has to be completed in 30 min. You also have to consider the time required to apply the changes to data. Table 36 gives an idea of the time required to apply changed pages to the table space or partition. From Table 36, you can conclude that updates in the range of 5% to 10% can be accommodated.

Table 36. Time to Apply Updated Pages to Table Space or Partition. The log file read time and image copy read time are not included.

Updated Pages (%)	1 GB (262144 pages)	2 GB (524288 pages)	4 GB (1048576 pages)
1	00:00:52	00:01:45	00:03:30
5	00:04:22	00:08:44	00:17:29
10	00:08:44	00:17:29	00:34:57
20	00:17:29	00:34:57	01:09:54
50	00:43:41	01:27:23	02:54:46
100	01:27:23	02:54:46	05:49:32

Note:

The assumption is that each page is updated only once

4.7 COPY

In this section we deal with the duration of an image copy to help you predict the time required. To help you understand the performance-sensitive variables, we present in Table 37 a formula to calculate elapsed times for image copy. The assumptions are explained in that table.

Table 37. Full and Incremental Image Copy with SHRLEVEL CHANGE: Elapsed Time Formulas

Database	Full Image Copy	Incremental Image Copy
DB2 V3	NP * 2 + NUP * 16	NUP * (20+16)
DB2 V4	NP * 1.8	NUP * (1.9 to 20)
DB2 V5	NP * 1.5	NUP * (1.6 to 20)

Note:

NP Number of pages copied
NUP Number of updated pages

- Assuming no I/O bottleneck
- If SHRLEVEL REFERENCE
  - DB2 V3 full image copy = NP * 2 + NUP * (2 to 16)
  - DB2 V3 incremental image copy = NUP * (20+2 to 16)
  - DB2 V4 and DB2 V5 same as SHRLEVEL CHANGE
- A range of numbers in incremental image copy formula to reflect the distance between pages copied (smaller number for pages close by)
- 9672 R_3 CPU time is 10% to 15% of elapsed time

In Table 37, for DB2 V4 and DB2 V5, the full image copy times are independent of the number of updated pages (NUP) because both these releases copy the pages without writing them back.

Table 38 compares the three releases of DB2 for full image copy with SHRLEVEL CHANGE. There is no difference in elapsed time between SHRLEVEL CHANGE and SHRLEVEL REFERENCE in DB2 V4 and DB2 V5.

Table 38. Full Image Copy with SHRLEVEL CHANGE: Elapsed Times in Milliseconds

Size of Table Space or Partition (GB)	DB2 V3	DB2 V4	DB2 V5
0.5	00:07:52	00:03:56	00:03:17
1	00:15:44	00:07:52	00:06:33
2	00:31:27	00:15:44	00:13:06
4	01:02:55	00:31:27	00:26:13

Note:

We make the following assumptions:

- No I/O bottleneck
- 10% updated pages

You can see a big difference in elapsed time between full image copy under DB2 V3 and DB2 V4 or DB2 V5. We assumed 10% of the pages were updated. When you change that rate to 20%, the numbers increase by almost 50%.

Table 39 shows the incremental incremental image copy times for DB2 V3. The columns refer to various percentages of updated pages. As it turns out under DB2 V3, for a table space or partition that has 10% or more updated pages, the full image copy is more advisable than an incremental image copy as it will not take more time. In addition it saves the merging of incremental image copies later.

Table 39. Incremental Image Copy with SHRLEVEL CHANGE: Elapsed Times in Milliseconds with DB2 V3.
Time is related to size and percentage of updated pages.

Size of Table Space or Partition (GB)	1%	10%	100%
0.5	00:04:43	00:07:52	00:39:19
1	00:09:26	00:15:44	01:18:39
2	00:18:52	00:31:27	02:37:17
4	00:37:45	01:02:55	05:14:34

Note: Assuming no I/O bottleneck

Now we look at the image copy times for DB2 V5. We ignore the calculations for DB2 V4 as they are not too different from DB2 V5 and in capacity planning terms can be considered identical. In DB2 V4 the CPU time and elapsed time for incremental image copy is in the 20% range. The formula given in Table 37 on page 85 for DB2 V5 incremental image copy contains a time factor of *1.6 to 20* milliseconds per I/O access. This factor needs interpretation. We used it as follows: Any updated page in the range of short seek is counted as 2 milliseconds, any other as 20 milliseconds. We assumed that about 30% of the pages were clustered in this way.

Table 40. Incremental Image Copy with SHRLEVEL CHANGE: Elapsed Times in Milliseconds with DB2 V5.
Time is related to size and percentage of updated pages.

Table Space Size (GB)	1%	10%	100%
0.5	00:00:19	00:03:10	00:03:30
1	00:00:38	00:06:20	00:06:59
2	00:01:16	00:12:39	00:13:59
4	00:02:32	00:25:18	00:27:58

Note:

- 30% updated pages clustered
- Assuming no I/O bottleneck
- Identical for SHRLEVEL REFERENCE and SHRLEVEL CHANGE

As in DB2 V3 we see that the 10% limit applies for the elapsed times of incremental image copy. On average we think that it is advisable to run a full image copy if the number of updated pages is higher than 10%. The image copy

utility parameter CHANGELIMIT(1,10) creates a full image copy when the number of changed pages is 10% or greater. That value is the default for DB2 V5.

The CHANGELIMIT option lets you tell DB2 whether it should make a copy based on the percentage of changed pages in a table space, partition, or data set. DB2 also reports the number of changed pages, which lets you monitor the rate of change of a particular object.

For DB2 Estimator these image copy times are comparable, so we do not present them here.

Chapter 5. Migration to a Data Sharing Environment

For data sharing, all of the considerations discussed in previous chapters are true and valid; they just need some refinement.

Changing a system to data sharing does not change most costs in the system. However, two major areas of cost are:

- Global locking—Propagating global locks beyond the local IRLM, global lock contention negotiation and resolution
- Data coherency—Page registration and deregistration, page refresh caused by cross-invalidation, force at commit protocol, and destage of changed data from coupling facility to disk storage

Capacity planning for applications running in a data sharing environment is a very similar process to that for applications running in a non-data-sharing environment except that the data sharing costs associated with global locking and data coherency should be added in. Capacity planning is required for coupling facility resources (CPU, processor storage, and coupling facility links). In the case of an existing environment the adjustments explained in this chapter can be used to come to a useable capacity prediction. In the sections that follow we explain how to approach the task best and which tools to use.

Disclaimer

There is much less experience estimating the CPU cost of data sharing compared to the traditional non-data-sharing environment. We provide a first-cut rule-of-thumb methodology here but stress that its accuracy is not at the same level as the non-data-sharing CPU cost estimation, because of a very limited measurement validation of the methodology used.

The numbers provided here are subject to change at any time as better estimates are obtained and to reflect the continuous product changes.

5.1 Methodology

The methods discussed here are kept simple and efficient. When we go from section to section we refine the methods, starting with a simple rule-of-thumb and ending with the tools we think are best for the task of addressing capacity planning questions for data sharing.

We have made the following assumptions that may have an impact on the validity of the formulas in your environment:

Environment

A well-tuned data sharing environment with no bottleneck in CF, link, group buffer pool, or lock structure. CPU times given relate to group-buffer-pool-dependent page sets or partitions.

Lock contention

Minimal lock contention (less than 1%).

If the lock contention is higher than 1%, DB2 V5 may help significantly, as its main focus is in reducing all types of data sharing

related lock contentions, including space map page P-lock contention, data page P-lock contention, type 2 index leaf page P-lock contention, XES contention, false hash contention, and index tree P-lock contention.

We also assume that the following authorized program analysis reports (APARs) are applied on your system:

PN85387 Changes the way DB2 locks a partitioned table space in DB2 V4; instead of locking each and every partition, locks only the last partition. This has the following performance benefits:

- Fewer IRLM lock, change, and unlock requests
- Less IRLM storage consumed
- Less IRLM latch contention
- Lower IRLM SRB time

In addition, in the data sharing environment the APAR changes have the following performance benefits:

- Fewer locks propagated to XES
- Fewer XES level global contentions

Fewer XES managed locks means peer recovery for a failed member takes less time.

The overall effect will vary depending on the amount of locking done on the partitions. For example, the impact will be substantial if there are a lot of partitions that are frequently accessed by plans that are bound with RELEASE(COMMIT).

The CPU reduction improvement (impact) could be very significant for transaction workloads that are relatively light in terms of SQL processing accessing heavily partitioned table space(s), and where either the application plans are bound with RELEASE(COMMIT) or there is effectively little or no thread reuse. With this APAR the performance benefits of binding with RELEASE(DEALLOCATE) are diluted when working with partitioned table spaces.

PN88155 For avoiding empty page reads in data sharing (DB2 V4)

PQ00496 For enabling register page list (RPL) in DB2 V4.
The RPL function provides feedback about the pages for which registration is being requested. This feedback includes an indication of whether the page is in the group buffer pool and, if so, whether it is clean or dirty. The feedback is used to decide whether to read the page from the group buffer pool or from DASD.

The RPL function is particularly useful for workloads with significant sequential access with use of prefetch. This function provides for batch registration and deregistration. MVS 5.2 plus maintenance and CFLEVEL=2 are prerequisites. After type 2 indexes and CURRENTDATA(NO), this was the next most significant factor in achieving performance goals for DB2 data sharing. It is worth 3%-4% in terms of reducing the overhead for IBM Relational Warehouse Workload (IRWW).

PQ02616 Reduced page P-locks
Page P-locks will no longer be acquired for type 2 index pages when the index's page set or partition P-lock is held in state SIX (single DB2 member updating the page set or partition, with other members

reading it), and the index has no repeatable read (RR) claimers on other members. To preserve the RR semantics, all updated index pages will be written out to the group buffer pool when the member switches from not acquiring the page P-locks to acquiring them. Additionally, an index page set or partition P-lock held in SIX mode will be retained in SIX mode instead of IX if there is a DB2 subsystem failure. These changes apply only to those indexes for which dynamic tracking of RR interest is performed, that is, all indexes other than those in the DB2 catalog and directory. For all indexes and table spaces whose page set or partition P-lock is held in SIX mode, the page P-lock will no longer be immediately released after the page is released or written to the group buffer pool. Instead, it will be kept (in anticipation of being needed again later) until the page buffer is stolen or requested by another member. Requests for the page P-lock from another member will be highly unlikely, unless there are RR claimers on the other member.

OW20060 For MVS/ESA SP Version 5 Release 2 for storage management enhancement in asynchronous data sharing process. The number of GETMAINs and FREEMAINs performed for SRB processing is reduced. These changes apply only to HBB5520 and above.

We assume that you use type 2 indexes. This is highly desirable because in the measurement of IRWW in DB2 V4, for example, the use of type 1 indexes instead of type 2 indexes doubled the cost of data sharing. See *DATABASE 2 for MVS/ESA Version 4 Data Sharing Performance Topics* (SG24-4611).

We also assume that you use CURRENTDATA(NO). The performance benefits with the combination of type 2 indexes and CURRENTDATA(NO) are significant.

Most of the information detailed here is taken from the *Data Sharing: Planning and Administration Guide* (SC26-8961) and *OS/390 MVS Parallel Sysplex Capacity Planning* (SG24-4680).

5.2 Data Sharing CPU Requirement

The data sharing CPU requirement has been established and published in the announcement letter of DB2 and in the *DATABASE 2 for MVS/ESA Version 4 Data Sharing Performance Topics* (SG24-4611).

Caution

We emphasize that this CPU requirement is only for the specific IRWW workload with type 2 indexes, page level locking, CURRENTDATA(NO), and a global lock contention rate of 1%. Data sharing overhead for other environments will vary. Most customers experience lower overhead with OLTP workloads.

5.3 Data Sharing IRLM Storage Requirement

For data sharing, plan for additional storage because of additional data sharing locks called P-locks. These locks are held on open page sets and on database descriptors (DBDs), skeleton cursor tables (SKCTs), and skeleton package tables (SKPTs). Unlike transaction locks, storage for P-locks is held even when there is no transaction activity, and therefore P-locks consume storage even with no transaction activity.

Plan also for extra storage that IRLM needs to build retained locks in case other members fail or you have to run IRLM diagnostic traces. The variables you need to account for are shown in Table 41.

Table 41. Variables Used to Estimate Additional IRLM Storage

Variable	Description	Calculation
X	P-locks	N = (MAX_OPEN_DATA_SETS × 500) X = N + (N × 0.40)
Note: The formula assumes that more than one P-lock might be held on a page set occasionally (such as for castout activity) and estimates about 40% for P-locks on the EDM pool objects and for short-lived page P-locks. If you know that your EDM pool has relatively few objects in it, you could use a lower percentage for that value. See Section 5 of the *DB2 for OS/390 Version 5 Administration Guide Volume 2* (SC26-8957) for more information about estimating the maximum number of open data sets, or use the value specified for the DSMAX subsystem parameter.		
Y	Ability to hold update retained locks for a failed member	Dependent on how update-intensive the workload is. Start with the following: Y = 0.25X
A	Storage for default diagnostic traces (applies also to non-data-sharing environment)	0.75 MB
B	Storage for extra diagnostic traces that you might run (applies also to non-data-sharing environment)	1.75 MB
I	Non-data-sharing IRLM storage	Storage required in your installation without data sharing
Formula: IRLM_STORAGE = **X** + **Y** + **A** + **B** + **I**		

For example, assume that your non-data-sharing IRLM storage estimate is 5 MB. If you estimate that this DB2 member could have as many as 8000 open data sets, you could calculate the IRLM storage as follows:

```
(8000 * 500) + 1600 KB = 5.60 MB
 +
1.40 MB (for retained locks)
 +
0.75 MB (default trace storage)
 +
1.75 MB (extra trace storage)
 +
5 MB (non-data-sharing estimate)
----------------------
Total IRLM storage = 14.50 MB
```

5.4 Increase IRLM Storage for Sysplex Query Parallelism

Sysplex query parallelism uses IRLM Notify messages for the following functions:

- The coordinator uses Notify messages to communicate with the assistants.
- The assistants use Notify messages to communicate with the parallelism coordinator.
- The main use of the Notify messages is to pass data between the assistants and the coordinator.

To use Sysplex query parallelism, make sure that you have enough storage specified on the MAXCSA parameter of the IRLM startup procedure to handle these messages.

MAXCSA is only applicable when PC=NO is selected. When adjusting MAXCSA, you must also make sure that the CSA parameter in IEASYSnn PARMLIB member is adjusted.

5.4.1 Calculating Storage for the Coordinator

Any member that can be a coordinator needs approximately 200 KB of extra storage for messages that the coordinator sends to the assistants. The amount of extra storage is dependent on the number of messages. 200 KB is a good starting point, and you must tune later according to the concurrent queries sent across the Sysplex.

5.4.2 Calculating Storage for the Assistants

To estimate the amount of extra storage required by IRLM on an assisting DB2 (any DB2 that receives query work from another DB2), estimate the number of parallel tasks that can run concurrently on the assisting DB2 and divide that by the number of members sharing the query work. Multiply the result by 32 KB to get an estimate of the amount of extra storage needed on the assistants.

$$\frac{\text{Number of queries} \times \text{Maximum concurrent tasks}}{\text{Number of members}} \times 32 \text{ KB}$$

For example, assume you have a data sharing group in which all four members participate in processing parallel queries. If you have at most 10 queries

executing concurrently, and the highest number of parallel tasks is approximately 40, the calculation is:

$$10 \times \frac{40}{4} = 100$$

$$100 \times 32\ \text{KB} = 3\ \text{MB of extra storage on the assistant}$$

To estimate the number of parallel tasks, you can look at EXPLAIN output or instrumentation records for the concurrent queries.

5.5 Sizing of Data Sharing Structures

In this section we deal first with sizing assessments and essentially summarize what is contained in the *Data Sharing: Planning and Administration Guide* (SC26-8961). DB2 has three types of structures inside a coupling facility: A single shared communication area (SCA), implemented as an MVS list structure; a single lock structure, implemented as such; and multiple group buffer pools, implemented as MVS cache structures.

Each virtual buffer pool in DB2 can have a corresponding group buffer pool. If the corresponding group buffer pool is not present, all table spaces and indexes using the specific virtual buffer pool cannot be opened for update by more than one member at any time. Thus simultaneous read is allowed, but no member can update the table spaces and indexes while another member is reading them.

Group buffer pool by default is a write-only cache and is not a replacement for hiperpool across members of a data sharing group. It is still important to maintain a high read hit ratio for each local buffer pool (virtual buffer pool+hiperpool). If there is less than 2 GB on a CMOS machine, it is better to define all processor storage as central storage, assign all hiperpool buffers to the virtual pool, and dispense with use of hiperpool. The only exception to this is where there is a strong requirement to use DFSORT or HiPerSort, which requires expanded storage, in which case you might decide to fence some of the processor storage as expanded storage even though the total amount of processor storage is less than or equal to 2 GB.

5.5.1 SCA Structure Size

The SCA is a list structure in the coupling facility. The size of the SCA is critical. However, the frequency of access to the SCA is not critical. The access frequency in DB2 V5 is 1 sec from each member for global commit log sequence number (GCLSN) tracking. However, there are other types of miscellaneous SCA accesses, so you would expect to see more than one request per second in the RMF coupling facility structure activity report for SCA. But access rates to SCA are generally very small compared to lock and group buffer pools. Most of the structure will remain empty most of the time. It is used for page set or partition exceptions when DB2 determines that parts of a page set or partition or a complete page set or partition are no longer accessible. Because much of the space in the SCA is taken up with exception information, you reclaim space by correcting database exception conditions. See Table 42 for preliminary sizing information.

Table 42. SCA Storage Sizes			
Site Size	**Databases**	**Tables**	**SCA Structure Size**
Small	50	500	8 MB
Medium	200	2000	16 MB
Large	400	4000	32 MB
Extra Large	600	6000	48 MB

Running out of space in the structure can cause DB2 to come down in case of exceptions while writing to page sets or partitions.

5.5.2 Lock Structure Size

The sizing assumes 100% data sharing. For installation planning purposes, we offer the following recommendation: Use a power of 2 for size. Typical sizes used are 16, 32, 64, and 128 MB. Start with 64 MB and move down or up based on amount of false contention. In the overall scheme of things, both SCA and LOCK1 are small.

5.5.3 Group Buffer Pool Structure Size

The group buffer pool is a cache structure in the coupling facility. By default the group buffer pool is used only if a page set or partition that belongs to that particular buffer pool is group-buffer-pool-dependent, that is, when at least one member of the data sharing group is updating and at the same time at least one other member is reading or updating. The size of the data pages is equal to the size of the corresponding virtual buffer pool, that is, either 4 KB or 32 KB. We assume that you specify GBPCACHE CHANGED on all table spaces, that is, only changed pages will go into the group buffer pool. See Table 43 for first estimates.

Table 43. DB2 Group Buffer Pool Sizing (GBPCACHE CHANGED Specified for All Table Spaces)	
Factor 1	**Condition**
10%	For very light sharing with low update activity 2
40%	For heavy sharing with high update activity 3

Note:

1 The sum of the related virtual buffer pools in all data sharing members is to be multiplied by this factor.

2 Minimum 10 MB advisable
Remember, a directory entry is about 208 bytes, that is, 5% of a page. If you want to register all pages in your virtual buffer pool and hiperpool, you will need exactly 5% of the size of your virtual buffer pool and hiperpool. Specifying 5% will therefore not allow all data pages in the virtual buffer pool and hiperpool to be registered. The default directory-to-data ratio of 5:1 will allow 20% of your data pages to be registered.

3 Needs more detailed evaluation

Be aware of the following considerations in order of importance:

1. Avoid directory reclaims causing cross-invalidation.
2. Avoid write failures.
3. Achieve high read hit ratios on group buffer pool synchronous reads following cross-invalidation.

The following example illustrates the formula in Table 43:

Example:

- Member 1 virtual buffer pool BP1 size is 200 MB
- Member 1 hiperpool size is 200 MB
- Member 2 virtual buffer pool BP1 size is 200 MB
- Member 2 hiperpool size is 200 MB
- Low update activity (or light data sharing)
- 10% * (400 + 400) <MB> = 80 <MB> for the GBP

Directory-to-Data Ratio: A simple formula to determine the directory-to-data ratio is:

$$\text{Directory-to-Data Ratio} = \frac{100}{\text{Factor}} + 1$$

Thus, for example, if you choose 10% and 40% for “Factor” as shown in Table 43, you would get a directory-to-data ratio of 11:1 and 3.5:1 respectively.

Recommendation

The IBM Dallas System Center recommendation for sizing group buffer pools to avoid directory reclaims is to make the group buffer pool size >= 32% of the summation of the member local buffer pools (including hiperpool). This assumes a 5:1 ratio. Some customers have experienced that this results in a group buffer pool size that is too large, and they are actually using a value < 32% for an initial sizing but allowing flexibility to dynamically expand the size if they need to. Note that in reality it is difficult to size the group buffer pool to guarantee no directory reclaims because there are some cases where DB2 does not deregister a page for XI. You should not get overly worried by directory entry reclaims unless there is a significant amount of cross invalidation caused by directory reclaims. Use the information provided by the -DIS GBPOOL command to identify this.

Storage Estimate for Caching All Data: For installation planning purposes, Table 44 provides initial estimates of the size of a DB2 group buffer pool when you want to cache read-only pages with changed pages (GBPCACHE ALL).

Table 44. DB2 Group Buffer Pool Sizing (GBPCACHE ALL Specified for All Table Spaces)

Factor	Condition
50%	For few table spaces, indexes, or partitions specify GBPCACHE ALL
100%	For many table spaces, indexes, or partitions, specify GBPCACHE ALL for a heavy sharing environment

For a more detailed discussion of sizing the group buffer pool, see 5.13, “Group Buffer Pool Structure” on page 118.

5.6 MVS Cross-System Coupling Facility Communication Volume

Now we take a look at MVS XCF and what considerations have to be made when extending MVS XCF to accommodate data sharing.

MVS XCF allows three methods for passing messages between members of the same MVS XCF group:

- Memory-to-memory: between members in a single MVS image
- Channel-to-channel (CTC) connections: between members on different OS/390 or MVS images
- CF list structure: between members on different OS/390 or MVS images

The method chosen is decided by MVS XCF based on the target of the message and the available signaling paths. In this discussion, we examine only the CTC and CF structure options because they are the methods most relevant for consideration when configuring the Parallel Sysplex.

When planning the signaling paths for MVS XCF to use, remember to ensure that there is redundancy in whatever you configure. For example, if you use CTCs for signaling, you define at least two paths into each system in the Parallel Sysplex and two paths out. This ensures that, if one path should fail, an alternative path will always be available. When using ESCON CTCs, you should also spread the paths over multiple ESCON Directors for availability reasons.

When using CF list structures exclusively for MVS XCF signaling (partial or no CTC connectivity between systems exists), define at least two signaling structures and allocate them in at least two CFs.

Recommendation: If two signaling structures are used and placed in two CFs, define at least two paths into each system in the Parallel Sysplex and two paths out. As a consequence, connectivity is maintained in case of problems with one structure.

Another good reason for having multiple signaling paths, either CTC or CF structure, is for throughput considerations. If MVS XCF has multiple paths, it will not try to put a request onto an already busy path. Alternatively, you can decide to route messages for specific MVS XCF groups along specific paths. This can be very useful if one particular group is known to have specific signaling requirements, such as it always uses long messages, or needs the signaling path at exactly the same time when one of the paths used by MVS XCF is busy with other activities such as locking by DB2. The system programmer could assign MVS XCF groups to transport classes and then associate the transport class with a specific signaling path or paths.

XCF messages are put into transport classes on the basis of either message size or XCF group name. A general recommendation is to route according to message size only with two transport classes to make best use of available storage for XCF buffers and to avoid overhead of dynamic buffer expansion and contraction.

5.7 Which Signaling Path Option Should You Use?

There are three possible options for the signaling between systems in a Parallel Sysplex:

- CTCs only
- CF structures only
- A combination of CTCs and CF structures

For low MVS XCF rates of activity, there is little performance difference between CTC and CF signaling paths. At high rates, performance is improved by using CTCs. MVS XCF will always use the fastest path for signaling, when there is a choice. The current recommendation therefore is to configure both CF structures and CTCs and allow MVS XCF to determine the best path.

The High Performance Coupling Links (HiPerLinks) available for G3 and G4 servers (S/390 Parallel Enterprise Servers) and corresponding CFs significantly improve the performance for CF signaling paths. For Sysplexes consisting of more than two CPCs, the performance is comparable with CTC connections. Therefore, with HiPerLinks it is possible to use CF-MVS XCF signaling paths and dispense with the added complexity and expense of CTCs.

Once the HiPerLink technology is installed on a CPC or CF, it cannot be combined with older links on that CPC or CF.

The additional MVS XCF traffic in data sharing is “caused” by lock contention. Contention is planned to be less than 1% of the locking rate. Each contention creates one message to the current lock owner and one or more messages originating from there to one or more members of the data sharing group.

Besides lock contention, communication is required, for example, for row locking and sharing of space map pages.

For high-level planning we recommend that for heavy data sharing you may need to add an additional link from each MVS to the CF (or CFs) that have MVS XCF structures.

This rule-of-thumb is valid for a planned 100% data sharing. For substantially less data sharing, deduct a proportional part of the MVS XCF communication volume. For example, when you plan to use only 10% data sharing, the existing MVS XCF communication capacity might be sufficient to cope with the additional load.

Plan for aggregate capacity for all users of XCF, not just DB2. A general recommendation is to use a combination of ESCON CTCs and CF list structures if the number of members in a group is less than 4. Beyond three members, use CF list structures only given the advent of HiPerLinks, as ESCON CTCs are difficult to administer and paths are point to point (unidirectional) and cannot be shared.

5.8 Coupling Facility CPU Requirement

Assume a Sysplex total capacity of X MIPS. Then, if the workload is mostly data sharing, the CF capacity requirement is 10% of X; if the workload has little data sharing, the requirement is 5% of X. This capacity requirement should be spread over two or more CFs. Furthermore, these rules-of-thumb are intended to keep the average CF CPU utilization below 50%. This provides good performance and room to handle the additional load in the event of a CF failure. For example, if a Sysplex has a total capacity of 1000 MIPS, and the workload is to be mostly data sharing, then it would need 100 MIPS of CF capacity which could be satisfied by configuring two 50 MIPS CFs. Actually, the 10% calculation includes enough extra capacity to allow the CFs to run at 50%. If you want to configure sufficient CF MIPS so that even in a CF failure situation you would run the remaining CF at less than 50%, you would need to have two CFs, each with a capacity equal to 10% (in this example, 100 MIPS) of Sysplex MIPS.

5.9 CPU Cost at the Host for Coupling Facility Interactions

CPU cost at the host for CF interactions are, for example, the time the CPU needs to communicate with the cache structure. Table 45 shows the CPU times in microseconds for some of the data sharing interactions between host and CF. The CPU times refer to the host CPU for some sample configurations of host and CF processor model. The times shown in Table 45 include both hardware and software costs associated with a CF interaction and represent additional CPU time in a data sharing environment compared to a non-data-sharing environment. So, P-lock time represents the total P-lock time, whereas L-lock time represents the total L-lock time minus the non-data-sharing L-lock time. P-locks occur only in a data sharing environment.

Table 45. CPU Times in Microseconds at the Host for Coupling Facility Interactions. Times for some sample configurations of host and coupling facility processor models are shown.

Activity	R1/C1 (G1)	R2/C2 (G2)	R3/C3 (G2)	R4/C4 (G3)	R5/C5 (G4)	H5/C4
Read no data (µs)	525	353	319	178	120	146
Read data, write, or castout (µs) 50 MB/sec link	600	428	394	253	195	221
... with 100 MB/sec link	560	388	354	213	155	181
Register page list (µs) 1	4155	2678	2398	1273	964	949
L-lock request (µs) 2	375	255	231	131	83	109
P-lock request (µs) 3	505	336	304	168	111	135

Note:

R1 for example, represents a 9672 R1 host processor
C1 for example, represents a 9674 C1 CF
G1 for example, represents R1 or C1, also called G1 processors
1 Register page list request assumes prefetch quantity of 32 and the cost is not affected by the number of pages in the group buffer pool.
2 Represents the total L-lock time minus the non-data-sharing L-lock time
3 Represents the total P-lock time. P-locks occur only in a data sharing environment.

Note

For combinations of hosts and CFs that are not explicitly given in Table 45, use the following method to estimate the times:

Approximately 75% of the delta between the times in any two columns is caused by the change in host, and approximately 25% is related to the change in CF. Thus, for example, to estimate the CPU time for an L-lock request of an R4 host connected to a C3 CF, you would take 75% from the R4/C4 time (75% of 131 µs, that is, 98.25) and add to this 25% from the R3/C3 time (25% of 231 µs, that is, 57.75), yielding 156 µs as the estimated R4/C3 time for this case.

5.10 Caching CPU Cost

The concept of “shared data” in DB2 is realized by using the CF. One of the structure types used is the group buffer pool. To estimate the cost for the group buffer pool we give a practical example in 5.12.1, “Applying the Data Sharing Cost Calculation” on page 107. Starting with this in a non-data-sharing environment, you can estimate group buffer pool activities in a data sharing environment. To do so we first explain what requests you have to take into consideration.

Look at Figure 23 on page 101, to see from where the counts can be taken.

```
BP5    READ OPERATIONS        QUANTITY   BP5    WRITE OPERATIONS       QUANTITY
-------------------------- -------- ---------------------------- --------
GETPAGE REQUEST              15157.00   BUFFER UPDATES             [3] 9106.00

GETPAGE REQUEST-SEQUENTIAL    4473.00   PAGES WRITTEN              [4] 4170.00
GETPAGE REQUEST-RANDOM       10684.00   BUFF.UPDATES/PAGES WRITTEN       2.18
                                        ..................          .......
SYNCHRONOUS READS          [1] 4310.00
SYNCHRON. READS-SEQUENTIAL    4178.00
SYNCHRON. READS-RANDOM         132.00

GETPAGE PER SYN.READ-RANDOM     80.94

SEQUENTIAL PREFETCH REQUEST   8554.00
SEQUENTIAL PREFETCH READS  [2]   0.00

PAGES READ VIA SEQ.PREFETCH [1]  0.00
S.PRF.PAGES READ/S.PRF.READ      N/C

LIST PREFETCH REQUESTS           0.00
LIST PREFETCH READS        [2]   0.00

PAGES READ VIA LIST PREFTCH [1]  0.00
L.PRF.PAGES READ/L.PRF.READ      N/C

DYNAMIC PREFETCH REQUESTED       0.00
DYNAMIC PREFETCH READS     [2]   0.00

PAGES READ VIA DYN.PREFETCH [1]  0.00
D.PRF.PAGES READ/D.PRF.READ      N/C
..................          .......
```

Figure 23. Virtual Buffer Pool Activities from DB2 PM Statistics Report

The group buffer pool is essentially used for the following actions:

Read and register This is the sum of the synchronous reads and pages read because of sequential, list, and dynamic prefetch requests [1]. You have to establish the group buffer pool hit rate H and compute H × [1]. The group buffer pool hit rate is established by building $\frac{G}{G+R}$,

where G is the number of pages found in the group buffer pool, and R is the number of pages read from DASD. If a page is not found in the virtual buffer pool or hiperpool, in a data sharing environment DB2 reads the group buffer pool first. If the page is not found in the group buffer pool, DB2 reads the page from DASD. Compute this number by building (1 - H) × [1], where H is the group buffer pool hit rate.

Note

H should be low for the non-cross-invalidated case as the group buffer pool by default is write-cache. However, H should be high for the cross-invalidated case in a well-tuned environment. The value of H depends on the application.

Register page list This is the sum of sequential, list, and dynamic prefetch reads 2. In prefetch mode, DB2 presents all pages to be read to the CF as a list to register those pages first. The CF responds with the pages available in the group buffer pool.

Write A page is written from the virtual buffer pool to the group buffer pool. This is done, for example, at COMMIT. Use a number between number of buffer updates 3 and pages written 4. The numbers reported in the Statistics Report apply to virtual buffer pool castout and are related to thresholds of the virtual buffer pool, whereas DB2 writes the pages to the group buffer pool at COMMIT at the latest.

Castout Castout is an asynchronous process in DB2 that reads pages from the group buffer pool to write them back to their page set or partition. You can use the number of pages written 4 assuming comparable thresholds in the group buffer pool as in the virtual buffer pool. Generally a number between number of buffer updates 3 and pages written 4 is good.

Other requests These are not further detailed here and are considered as additional overhead.

Only statistics from update-shared data and indexes should be included in the calculation of data sharing cost; that is, statistics, for example, on read-only data, nonshared update data, and work files should be excluded.

Table 46 on page 103 shows how to calculate the number of group buffer pool requests.

Table 46. Calculating the Number of Group Buffer Pool Requests	
Activity	**Formula**
Read no data	Number of synchronous reads + Getpage no read 1
Read data	Number of prefetch requests × 5 + XI + updated pages not found in virtual buffer pool but found in group buffer pool (especially for large group buffer pools) 2
Write	Number of pages written × 1.2 3
Castout	Number of pages written
Register page list	Number of prefetch requests 4
Other requests 5	Add 10%

Note:

1 Getpage no read = Buffer Manager Getpage no read for insert to an empty page at the end of data set with DB2 V4 APAR PN88155. Use 0 if you have no idea and instead add an additional 10% contingency to item 5 to represent "Read no data." We do not think there is any way to determine the number of Getpage no reads from Statistics.

2 5 is the assumed average number of pages found in the group buffer pool in an average prefetch request. Adjust this number as needed for a given environment.

$$XI = \text{Cross invalidation cost} = \text{Buffer Update} \times 0.2 \times \frac{(M - 1)}{M},$$

where M = number of data sharing members, and 0.2 represents the assumption that only 20% of the cross-invalidated data will be reused.

Use 0 if you have no idea and instead add an additional 10% contingency to item 5 to represent "Read data."

3 Assume 20% more pages written to the group buffer pool than to DASD because a group buffer pool write is forced at commit, whereas DASD write is done when write thresholds are reached.

4 With DB2 V4 APAR PQ00496

5.11 Locking CPU Cost

Locking CPU cost is the time needed to communicate with the lock structure. Only statistics from shared-update data and indexes should be included in the calculation of data sharing cost; that is, statistics, for example, on read-only data, non-shared update data, and work files should be excluded.

For L-locks, both parent (table space or partition) and child (page or row) locks should be included, but for P-locks, only the page P-locks should be included, as the parent P-locks are considered to be negligible.

Table 47 on page 104 shows how to calculate the number of lock requests to the CF.

Table 47. Calculating the Number of Lock Requests

Activity	Formula
L-lock request	Number of L-lock requests × 2 + number of change requests 1
P-lock request 6	NIP for type 2 index page P-lock × 2 2 + NIDU for space map page P-lock × 0.2 3 + NDP for data page P-lock × 2 4
Contingency	Add 10% to 20% 5

Note:

.

1 Some L-lock requests need not be propagated and a page accessed by multiple concurrent transactions needs to be propagated only once. Yet lock avoidance is not as effective in a data sharing environment. Therefore, assume these offset each other and the number of L-lock requests is the same in data sharing and non-data-sharing environments.

Justification

If you take the IRWW measurement as an example (and this is a very well tuned case), the following statistics are indicated:

	<------ L-lock Requests ------>	
	Issued	Propagated
Lock	6.84	5.58
Unlock	1.61	0.87
Change	3.78	3.77
Total per transaction	12.23	10.22

This results in 16% difference. However, the total number of IRLM requests in non-data-sharing environment can be smaller because of more effective lock avoidance. Since we are trying to estimate the data sharing overhead from non-data-sharing statistics available, assuming the number of propagated requests close to the number of IRLM requests in non-data-sharing environment appears to be a good assumption and a straightforward and reasonable approach.

2 NIP = Number of index pages updated in insert, delete, or update

3 NIDU = Number of insert, delete, or update calls.
Adjust upward if a single SQL call does multi-row operations.

4 NDP = Number of data pages updated in Insert, Delete, or Update assuming row locking is used.

5 Includes small contention, if any

6 If only one updating member (that is, cache state of SIX) and no RR readers

- No type 2 index leaf page P-lock request (DB2 V4 APAR PQ02616)
- Updater's page L-locks are not propagated to coupling facility if UR readers
- Still group-buffer-pool-dependent and therefore no change to group buffer pool activities

For reference we include a DB2 PM Locking Statistics Report in B.2, “DB2 PM Statistics Report: Locking Activity” on page 139. Figure 24 is an excerpt of the report.

```
LOCKING ACTIVITY             QUANTITY   DATA SHARING LOCKING         QUANTITY
---------------------------  --------   --------------------------  --------
 . . . . . . . . . . . . . .  . . . .   LOCK REQUESTS (P-LOCKS)      25890.00
LOCK REQUESTS     1            147.5K   UNLOCK REQUESTS (P-LOCKS)    25855.00
UNLOCK REQUESTS              54317.00   CHANGE REQUESTS (P-LOCKS)       10.00
QUERY REQUESTS                   0.00
CHANGE REQUESTS              65204.00   SYNCH.XES - LOCK REQUESTS  2   128.9K
OTHER REQUESTS                   0.00   SYNCH.XES - CHANGE REQUESTS  64952.00
                                        SYNCH.XES - UNLOCK REQUESTS  3 119.6K
                                        ASYNCH.XES - RESOURCES           0.00
```

Figure 24. Locking Activities from Statistics Report

If you compare 1, 2, and 3 in Figure 24, our assumption made for lock requests is justified.

Note

Sections 5.10, “Caching CPU Cost” on page 100 and 5.11, “Locking CPU Cost” on page 103 assume 100% data sharing. The degree of data sharing (between 0.00 and 1.00) must be applied to the locking and caching CPU costs. To collect data in a non-data-sharing environment to help determine data sharing costs, it is highly desirable to use buffer pool isolation to get better information out of DB2 Statistics trace data. At the first level, you should separate out the catalog and directory from work files, Read/Write shared data, and nonshared data. Within shared data, you should separate out indexes from data.

5.12 Data Sharing Cost Calculation

In this section we provide the formula used to define data sharing cost. Introducing data sharing into an existing non-data-sharing environment brings quantifiable benefits; however, there are costs attached to it as well. See Figure 25 on page 106.

$$\text{Data Sharing Cost} = \frac{\text{Locking cost} + \text{Caching cost}}{\text{Average transaction CPU time}}$$

This is the additional percentage CPU cost for transactions in a data sharing environment.

Note:

Locking cost	Lock cost × Average number of lock requests per transaction
Caching cost	Group buffer pool cost × Average number of group buffer pool requests per transaction
Average transaction CPU time	You should not consider Accounting TCB time only; it can be too small. Instead, consider adding to this time the prorated TCB and SRB times for the DB2 address spaces from Statistics. An alternative to this is to use the data from RMF CPU activity reports.

Figure 25. Data Sharing Cost

The lock costs and the group buffer pool costs typically vary between 100 and 600 µs depending on the host processor speed, CF processor speed, link speed, and type of CF requests.

Example

The example demonstrates the application of the data sharing cost formula given in Figure 25. The data sharing cost in this example is the additional DB2 CPU. The data sharing overhead at the application and total system levels may be much smaller.

We make the following assumptions:

- Number of L-locks 30
- Number of P-locks 16
- Number of pages written .. 20
- Number of castout pages .. 20

We derive the following values from the last column in Table 45:

- CPU time for L-lock = $\frac{109}{1000}$ = 0.11 ms
- CPU time for P-lock = $\frac{135}{1000}$ = 0.14 ms
- CPU time for write = $\frac{181}{1000}$ = 0.18 ms
- CPU time for castout = $\frac{181}{1000}$ = 0.18 ms

We assume an H5 host, calculate the lock cost and the group buffer pool cost, and then apply the formula to get the data sharing cost:

- A profile of an average transaction currently running on an H5 host in a non-data-sharing environment:
  - 80 ms accounting class 2 TCB time
  - 20 ms DB2 and IRLM SRB time prorated
  - 100 ms non-DB2 time
- Total lock cost = $1.2 \times 5.5 = 7$ ms, where multiplier 1.2 accounts for 20% contingency (from **5** in Table 47 on page 104), consisting of:
  - L-lock = $30 \times 0.11 = 3.3$ ms
  - P-lock = $16 \times 0.14 = 2.2$ ms
- Total GBP cost = $1.3 \times 7.9 = 10$ ms, where multiplier 1.3 accounts for 30% contingency (10% each for read no data, read data, and other requests, from **1**, **2**, and **5** in Table 46 on page 103), consisting of:
  - Write = $20 \times 0.18 \times 1.2 = 4.3$ ms
  - Castout = $20 \times 0.18 = 3.6$ ms
- Total data sharing cost = 17 ms or 12% on an H5 host and C4 CF

5.12.1 Applying the Data Sharing Cost Calculation

In this section we relate the data sharing cost formulas to SQL requests and attach to each SQL statement used in our transaction the cost for the CF. Having demonstrated the principle we now apply it to the application layout as presented in 3.3.3, “Sample Transaction Detail” on page 59.

First let us list the assumptions we made:

- Hardware
  - MVS CP processor model 9672-R5
  - G4 CF, level 02 or higher
  - 100 MB/sec links
    The slower 50 MB/sec links increase the data sharing overhead by 0.5 to 1.2%.
- DB2
  - 100% data sharing
    All calculations are based on 100%. A lower percentage changes the data sharing cost proportionally.
  - DB2 data sharing system is in steady state, that is, no excessive data set opening or closing. Frequent opening and closing increase the coupling facility interactions and add considerable load to the MVS XCF communication.
  - Virtual buffer pool and hiperpool have a hit rate of about 80%. Changing the hit rate by 10% changes the data sharing overhead by 1%; that is, when you have a hit rate of 70% instead of 80%, the calculated data sharing overhead of our example of 11% goes to 12%.
  - Read group buffer pool with no data returned about 20%, 80% found in virtual buffer pool
    This statement is related to the previous point. We assume the worst case where you do check the group buffer pool and have to read the required data from DASD.
  - Cross-system invalidation (XI) rates are comparable for all members, so XI can be assumed for each UPDATE, INSERT, and DELETE. Symmetric processes are assumed on all members of the data sharing group. Essentially an XI reduces the buffer pool hit rate.
  - Castout will move about 80% of the pages written to the group buffer pool. This is to reduce the castout overhead. The sensitivity of this parameter is that a change of 20% will affect a change of about 1% of the data sharing overhead; that is, when you change this parameter from 80% to 100%, the data sharing overhead will increase by 1%. However, causing more I/O to occur can adversely affect your system, for example, in response time and predictability.
  - We ignored the locking and MVS XCF communication required, assuming thread reuse and steady state for the system.
  - Lock avoidance 90%
    When you lower lock avoidance to 70%, the data sharing cost increases by 1% in our example. No lock avoidance pushes the data sharing costs up 3% in our example.
  - Open cursor without prefetch
    Our example does not include prefetch. So the example does not use the improved registering times for the RPL.
  - On INSERT and UPDATE 10% space map page update
    Changing this parameter in our example to 50% increases the data sharing overhead by less than 1%.
  - On INSERT and UPDATE no page split
    We considered this event a relatively infrequent action and did not calculate any values for it.

- Application
  - Upper bound and lower bound SQL statements are not different as far as their data sharing interactions with the coupling facility are concerned. Upper bound and lower bound apply only to the size of the page set or partitions, not to the way they are being used.
  - SELECT uses a single index
    Our example is based on probe type SELECTs that specify one valid key for the SELECT and do not scan index or table space.
  - Cursor stability used
    Cursor stability is needed for lock avoidance, and we think that it can be used in a majority of applications.
  - Close cursor removes all application locks, as cursor with hold is not used.
  - BIND with CURRENTDATA(NO)
  - BIND with RELEASE(DEALLOCATE)
    The alternative would have been RELEASE(COMMIT). We did not calculate the effects of RELEASE(COMMIT).
  - Update uses cursor
    Therefore update does not require that the page or index be read again. As we stated above, SELECT uses a single index, so an update has to read the second index only, if at all.
  - Index update probability 5%
    Increasing that probability to 25% increases our data sharing overhead by 1%.
  - Thread reuse
    We ignored the locking and MVS XCF activities associated with create thread and connect. Thread reuse can be determined by looking into the Accounting Report of DB2 PM. The fields NEW USER and RESIGNON under NORM TERM should be greater than zero to indicate thread reuse.
  - Sorting is not done for OPEN CURSOR.
    Access to the data is direct using the key on a single index.
  - Average number of rows per page is 15.
    This parameter influences locking. Changing it to 1, that is, each page contains one row, increases the data sharing overhead in our example by 1%.
  - Average number of indexes on each table is two type 2 indexes.
    Adding another two indexes in our example adds about 3% to the data sharing overhead. For an additional six indexes, it is 9% more overhead.

Having made these assumptions, we need to define the two terms “locking cost” and “caching cost” for the SQL statements as used in Table 14 on page 60. To do this, we use Table 48 on page 110, which contains the SQL statement types as used in Table 14 on page 60, with one difference: the distinction of lower bound and upper bound is not needed here as both apply to the same type of action. The difference between lower bound and upper bound is that the sizes of the tables differ. However, a difference in size does not affect coupling facility interactions. Table 48 defines the locking cost and caching cost.

Table 48. Formulas for Coupling Facility Interactions.
These formulas are related to SQL requests in our sample application.

Action	Locking Cost		Caching Cost		
	P-Lock	L-Lock	GBP_R	GBP_W	Castout
SELECT		0.2	$(1 + 1) \times 0.2$		
OPEN CURSOR		0.2	$(1 + 1) \times 0.2$		
FETCH		$\frac{0.2}{15}$	$\frac{0.2}{15}$		
INSERT	2 + 0.2	1	$(1 + 2) \times 0.2$	1 + 2	$(1 + 2) \times 0.8$
UPDATE	2 + 0.1	2	$(2 - 1) \times 0.05 \times 0.2$	1 + 0.1	$(1 + 2 \times 0.05) \times 0.8$
DELETE	2 + 0.2	1	$(1 + 2) \times 0.2$	1 + 2	$(1 + 2) \times 0.8$

Note:

Heading	Meaning
P-Lock	P-lock in lock structure
L-Lock	L-lock in lock structure
GBP_R	Read no data from group buffer pool, 20% not in virtual buffer pool
GBP_W	Data page write, read data, castout to or from group buffer pool
Castout	Castout from group buffer pool to DASD, 20% buffer reuse

Locking Cost

Expression	Meaning
0.2	10% data pages locked and unlocked, locking reduced by lock avoidance (SELECT, OPEN CURSOR, and FETCH)
15	Average number of rows in a page (FETCH)
2 + 0.2	Lock and unlock requests for each index and a 10% probability to lock and unlock the space map page of the data pages (INSERT and DELETE)
2 + 0.1	Lock and unlock requests for each index and a 5% probability to lock and unlock the space map page of the data pages (UPDATE)
1	Request for one X-lock (INSERT and DELETE)
2	Request for one U-lock and one X-lock (UPDATE)

Caching Cost

Expression	Meaning
1 + 1	Select one data page using one index, prerequisite for UPDATE
0.2	At 80% virtual buffer pool hit rate, data read 20% of the time.
15	Average number of rows in a page
1 + 2	Changes to one data page and two index pages
2 − 1	Read remaining indexes for UPDATE, one index already in storage because of SELECT
1+0.1	Writing one data page and one space map page
0.05	5% update probability for index
0.8	Castout requests are equal to about 80% of the number of write requests

Table 48 serves as explanation for the factors applied to the data sharing cost.

You could now use the formulas given in Table 48 and apply them to the SQL content of your system, as can be found in the DB2 PM Statistics Report. We have added an example in B.3, “DB2 PM Statistics Report: SQL Activity” on page 140.

Now that we have established the locking cost and the caching cost per statement, we can apply those costs to transactions as defined in 3.3.1.2, “Average Transaction Cost” on page 55. For that purpose we use the Table 14 on page 60. This table contains three parts, one for small transactions, one for medium transactions, and one for large transactions. Each of these parts forms the left side in the next three tables (Table 49, Table 50, Table 51).

All SQL statements considered in Table 14 on page 60 have a lower and an upper bound. As upper bound and lower bound SQL statements apply only to the size of the tables, and not to the interactions with the coupling facility in data sharing, we have simplified the list by listing only the SQL statement. For example, the coupling facility interactions listed for SELECT then apply to SELECT upper bound and to SELECT lower bound in Table 14 on page 60.

We calculate the data sharing costs for each transaction type separately. Table 49 gives you the data sharing costs for the small transactions.

Table 49. Data Sharing Cost for Small Sample Transactions.
CPU times are in milliseconds for IBM 9672-R15.

SQL Statement Type	Small		Coupling Facility Calls				
			Locking		Caching		
	# of Calls [1]	DB2 CPU (ms)	P – L [2]	L – L [3]	GBP Read [4]	GBP Write [5]	C.O. [6]
Create Thrd & Commit	1	2.0					
Select - LB	3	1.2		0.60	1.20		
Select - UB	3	2.4		0.60	1.20		
Open,fetch,close - LB	1	0.6		0.20	0.40		
Add'l fetch - LB	8	0.8		0.11	0.11		
Open,fetch,close - UB							
Add'l fetch - UB							
Update - LB	1	0.5	2.10	2.00	0.01	1.10	0.88
Update - UB							
Insert/delete - LB							
Insert/Delete - UB							
Sum CF interactions			2.10	3.51	2.92	1.10	0.88
CPU time for single CF interaction (µs) [7]			131	103	140	175	175
CPU CF interactions (ms)			0.28	0.36	0.41	0.19	0.15
Total SQL calls	17						
Total DB2 CPU		7.5					
Transaction CPU [8]		15.0					
Data sharing CPU %	9.27 [9]						

Note:

Small sample transaction, as defined in Table 14 on page 60

[2] P-lock request, [1] times CF interactions from Table 48 on page 110

[3] L-lock request, [1] times CF interactions from Table 48 on page 110

[4] Read no data, [1] times CF interactions from Table 48 on page 110

[5] Read data or WRITE, [1] times CF interactions from Table 48 on page 110

[6] Castout request, [1] times CF interactions from Table 48 on page 110

[7] CF interaction host CPU times, refer to Table 45 on page 100.

[8] Cost of total transaction, including DB2 cost. Assume that transaction manager and application cost is same as the DB2 cost (validate against your applications)

[9] = ∑Total DB2 CPU for CF interactions / Transaction CPU
= (0.28 + 0.36 + 0.41 + 0.19 + 0.15)/15.0

As you can see in Table 49, the transaction related data sharing overhead is in the range of 9%. The major reason behind this number is the relatively low content of INSERT, UPDATE, and DELETE activities.

Now let us look at the medium sized transactions in Table 50 on page 114. This tables reflects the data sharing activities for the transaction defined in Table 14 on page 60 as medium.

Table 50. Data Sharing Cost for Medium Sample Transactions. CPU times are in milliseconds for IBM 9672-R15.

SQL Statement Type	Medium		Coupling Facility Calls				
			Locking		Caching		
	# of Calls 1	DB2 CPU (ms)	P – L 2	L – L 3	GBP Read 4	GBP Write 5	C.O. 6
Create Thrd & Commit	1	2.5					
Select - LB	6	2.4		1.20	2.40		
Select - UB	12	9.6		2.40	4.80		
Open,fetch,close - LB	1	0.6		0.20	0.40		
Add'l fetch - LB	10	1.0		0.13	0.13		
Open,fetch,close - UB	1	1.0		0.20	0.40		
Add'l fetch - UB	10	2.0		0.13	0.13		
Update - LB	1	0.5	2.10	2.00	0.01	1.10	0.88
Update - UB	1	1.0	2.10	2.00	0.01	1.10	0.88
Insert/Delete - LB	1	0.8	2.20	1.00	0.60	3.00	2.40
Insert/Delete - UB	1	1.6	2.20	1.00	0.60	3.00	2.40
Sum CF interactions			8.60	10.26	9.48	8.20	6.56
CPU time for single CF interaction (µs) 7			131	103	140	175	175
CPU CF interactions (ms)			1.13	1.06	1.33	1.44	1.15
Total SQL calls	45						
Total DB2 CPU		23.0					
Transaction CPU 8		46.0					
Data sharing CPU %	13.28 9						

Note:

Medium sample transaction, as defined in Table 14 on page 60

2 P-lock request, 1 times 0.05 CF interactions from Table 48 on page 110

3 L-lock request, 1 times CF interactions from Table 48 on page 110

4 Read no data, 1 times CF interactions from Table 48 on page 110

5 Read data or WRITE, 1 times CF interactions from Table 48 on page 110

6 Castout request, 1 times CF interactions from Table 48 on page 110

7 CF interaction host CPU times, refer to Table 45 on page 100.

8 Cost of total transaction, including DB2 cost. Assume that transaction manager and application cost is same as the DB2 cost (validate against your applications)

9 = ∑Total DB2 CPU for CF interactions / Transaction CPU
= (1.13 + 1.06 + 1.33 + 1.44 + 1.15)/46.0

The transaction-related data sharing percentage for the medium sized transaction is in the range of 13%. Relating this to the total system capacity would lower that percentage to less than 10%.

When you compare the total locking calls (18.86 = 8.60 + 10.26 on average) with the total caching calls (24.24 = 9.48 + 8.20 + 6.56), you find that the caching calls are about 22% more than the locking calls $\frac{(24.24 - 18.86)}{24.24} \times 100 = 22$.

This gives you an idea of where to look for coupling facility link balancing.

Having no lock avoidance would change the relationship of caching calls to locking calls from 1.3:1 to 0.4:1. This represents an increase in locking of 300%.

Table 51 details the data sharing costs for the large transactions as defined in Table 14 on page 60.

Table 51. Data Sharing Cost for Large Sample Transactions.
CPU times are in milliseconds for IBM 9672-R15.

SQL Statement Type	Large		Coupling Facility Calls				
			Locking		Caching		
	# of Calls 1	DB2 CPU (ms)	P – L 2	L – L 3	GBP Read 4	GBP Write 5	C.O. 6
Create Thrd & Commit	1	3.0					
Select - LB	5	2.0		1.00	2.00		
Select - UB	15	12.0		3.00	6.00		
Open,fetch,close - LB	1	0.6		0.20	0.40		
Add'l fetch - LB	160	16.0		2.13	2.13		
Open,fetch,close - UB	1	1.0		0.20	0.40		
Add'l fetch - UB	160	32.0		2.13	2.13		
Update - LB	4	2.0	8.40	8.00	0.04	4.40	3.52
Update - UB	4	4.0	8.40	8.00	0.04	4.40	3.52
Insert/Delete - LB	6	4.8	13.20	6.00	3.60	18.00	14.40
Insert/Delete - UB	6	9.6	13.20	6.00	3.60	18.00	14.40
Sum CF interactions			43.20	36.66	20.34	44.80	35.84
CPU time for single CF interaction (µs) 7			131	103	140	175	175
CPU CF interactions (ms)			5.66	3.78	2.85	7.84	6.27
Total SQL calls	363						
Total DB2 CPU		87.0					
Transaction CPU 8		174.0					
Data sharing CPU %	15.17 9						

Note:

Large sample transaction, as defined in Table 14 on page 60

2 P-lock request, 1 times 0.05 CF interactions from Table 48 on page 110

3 L-lock request, 1 times CF interactions from Table 48 on page 110

4 Read no data, 1 times CF interactions from Table 48 on page 110

5 Read data or WRITE, 1 times CF interactions from Table 48 on page 110

6 Castout request, 1 times CF interactions from Table 48 on page 110

7 CF interaction host CPU times, refer to Table 45 on page 100.

8 Cost of total transaction, including DB2 cost. Assume that transaction manager and application cost is same as the DB2 cost (validate against your applications)

9 = ∑Total DB2 CPU for CF Calls / Transaction CPU
= (5.66 + 3.78 + 2.85 + 7.84 + 6.27)/174.0

This large transaction is defined by relatively heavy UPDATE, INSERT and DELETE activities. That entails heavier locking. Therefore the sensitivity to lock avoidance is reduced. Setting lock avoidance to zero results for our transaction in less than double the locking activity.

In Table 52 we present the ratio of cache activity to lock activity related to lock avoidance for our tables.

Table 52. Ratio of Cache Activity to Lock Activity Related to Lock Avoidance. The table shows the effect of 90% lock avoidance on lock structure calls in relation to the group buffer pool.

Transaction	Lock Avoidance	
	90%	**0%**
Small	0.9	0.3
Medium	1.3	0.4
Large	1.3	1.0

Note: Small transactions make fewer changes to data than medium and large transactions in our example. The effect of lock avoidance is more if the percentage of INSERT, UPDATE, and DELETE of a transaction is low.

To verify lock avoidance in your data sharing environment, look into the DB2 PM Statistics Report and compare the sum of all group buffer pool activities under the heading “GROUP TOT4K” in B.4, “DB2 PM Statistics Report: Group Buffer Pool Totals” on page 141 with the sum of all data sharing locking activities under the heading “DATA SHARING LOCKING” in B.2, “DB2 PM Statistics Report: Locking Activity” on page 139. You should find values resulting in ratios similar to the ratios in Table 52.

Our example shows very good lock avoidance.

Lock avoidance is determined based on log record sequence number (LRSN) or relative byte address (RBA) in page in relation to GCLSN value and on the possibly uncommitted (PUNC) bits in index RIDs and data page records. The GCLSN value is equal to the LRSN corresponding to the begin-UR record for the oldest outstanding UR in the data sharing group. For non-group-buffer-pool-dependent objects, the page set or partition value is used, just as in the non-data-sharing environment.

To achieve good lock avoidance you must avoid long-running jobs with update intent that do not take intermediate commit points.

5.12.2 Data Sharing Cost for Sample Application

As a final step we now apply the data sharing cost found for our sample transactions to our sample applications. We repeat the information from Table 11 on page 58 for the definition of the sample application in Table 53, and add the average data sharing overhead for our sample transaction and our sample application at 50% CPU utilization by DB2.

Table 53. Data Sharing Overhead for Our Application.
Applying the data sharing overhead of our transactions to our application.

Application	Transaction %			Application Data Sharing Overhead % 5
	Small 1	Medium 2	Large 3	
Light %	80	15	5	10.2
Medium %	15	80	5	12.8
Heavy %	15	45	40	13.5
Transaction data sharing overhead % 4	9.3	13.3	15.2	

Note:

- 5 = 1 × 4 + 2 × 4 + 3 × 4
- 5 : Percentage applies to 50% CPU utilization by DB2.

5.13 Group Buffer Pool Structure

A DB2 system can have multiple virtual buffer pools. In a data sharing environment, each virtual buffer pool can have an associated group buffer pool. A group buffer pool is a prerequisite to allow data sharing for any of the table spaces that relate to a particular virtual buffer pool.

Each group buffer pool has two parts: one for data pages and one for directory entries. The calculation in 5.5.3, “Group Buffer Pool Structure Size” on page 95 does not reflect that division. We now look into it in more detail to define not only the sizes but also the ratio of directory entries to data pages in the group buffer pool.

The approach here is different from that in 5.5.3, “Group Buffer Pool Structure Size” on page 95 where we have defined 5% as containing directory and data and did not care about the ratio of the directory and data entries. However, in this section we define 5% as data pages only and will then add later the required directory entries to the group buffer pool size.

5.13.1 Data Pages

We need to estimate the number of pages needed in the group buffer pool and how long they stay in the group buffer pool before being written out to the page set or partitions. Assume that the DB2 catalog is always group-buffer-pool-dependent. The size of the group buffer pool is dependent on the sum of the related group buffer pools in all members of the data sharing group and on the number of updated pages in data sharing mode.

5.13.2 Directory Entries

A directory entry contains control information for one database page, no matter in how many places that page is cached. For example, if page P1 is cached in the group buffer pool and in the virtual buffer pool of three members, that page has only one directory entry. That is the same for a page that is cached only in the hiperpool of a single member.

Each directory entry is 208 bytes for 4 KB pages and 264 bytes for 32 KB pages, assuming CFLEVEL=0. One entry is needed for each page used in a group-buffer-pool-dependent page set or partition. As the group buffer pool potentially can contain pages not in any virtual buffer pool or hiperpool, the maximum number of entries in the directory can be the sum of all related virtual buffer pools, hiperpools, and the group buffer pool pages.

5.13.3 Group Buffer Pool Sizing with Measurement Data

We provide in this section a formula to estimate the required group buffer pool size. It is based on an estimate of the desired residency time of an updated page in the group buffer pool and on the estimated percentage of data sharing of those pages. Anything else is measured or set data. The idea is that each member in its peak time has a specific page update rate.

GBP sizing formula for GBPCACHE CHANGED

Data_entries $U \times D \times R$

Data(MB) $\text{Data\_entries} \times \frac{P}{1024}$

Directory_entries $\text{Data\_entries} + (U \times (HP + VP))$

Directory(MB) $1.1 \times \text{Directory\_entries} \times \frac{0.2}{1024}$

GBP(MB) Data(MB) + Directory(MB)

RATIO $\frac{\text{Directory\_entries}}{\text{Data\_entries}}$

U Value between 0 and 1 representing degree of data sharing

Here is a formula to derive U from DB2 statistics if you already have a data sharing environment. The idea behind this formula is that we can define U by estimating the percentage of getpages that are issued for inter-DB2 read/write shared objects.

$$U = \text{Min}\ \frac{(((S + A) \times 1.1), G)}{G}$$

G Number of getpage requests

S Number of synchronous group buffer pool read requests. This includes both pages that are “found” and “not found” in the group buffer pool.

A Number of pages registered to the group buffer pool through prefetch. This includes both pages that are “found” and “not found” in the group buffer pool.

1.1 Accounts for the getpages issued for shared data which find the page in the buffer pool (“buffer pool hit”) and not cross-invalidated (getpage requests for shared data that require no group buffer pool interaction).

If you do not have a data sharing environment, you can estimate the degree of data sharing based on your knowledge of how the work will be split across the members and which objects will be inter-DB2 read/write shared. If you have no idea, use U=1 (100% data sharing).

D Number of pages written to DASD per second for all members, peak rate

R Average page residency time in the group buffer pool, in seconds. This value is application dependent, but its typical range may be assumed to be 30 to 180 sec. In general, you would want this value to be high enough so that when a changed page is written to the group buffer pool and invalidates local copies of the page in other DB2 members, the page remains resident in the group buffer pool long enough so that when the other members rereference their local copy of the cross-invalidated page, the page can be refreshed from the group buffer pool.

P Page size (4 KB or 32 KB)

HP Number of pages in the hiperpool for all members

VP Number of pages in the virtual buffer pool for all members

0.2 Size of a group buffer pool directory entry, in KB

1.1 Accounts for CF control storage

Group Buffer Pool Activity Block in DB2 PM V4 Statistics

SYN.READS(XI)-DATA RETURNED 1
SYN.READS(XI)-R/W INTEREST 2
SYN.READS(XI)-NO R/W INTER. 3
SYN.READS(NF)-DATA RETURNED 4
SYN.READS(NF)-R/W INTEREST 5
SYN.READS(NF)-NO R/W INTER. 6
ASYNC.READS-DATA RETURNED 7
ASYNC.READS-R/W INTEREST 8
ASYNC.READS-NO R/W INTEREST 9

CLEAN PAGES SYNC.WRITTEN
CHANGED PAGES SYNC.WRITTEN
CLEAN PAGES ASYNC.WRITTEN
CHANGED PAGES ASYNC.WRITTEN

PAGES CASTOUT
CASTOUT CLASS THRESHOLD
GROUP BP CASTOUT THRESHOLD
CASTOUT ENGINE NOT AVAIL.

WRITE ENGINE NOT AVAILABLE
READ FAILED-NO STORAGE
WRITE FAILED-NO STORAGE

OTHER REQUESTS

Note:

S in the formula to derive U for the group buffer pool size is the sum of fields 1, 2, 3, 4, 5, and 6.

A in the formula to derive U for the group buffer pool size is the sum of fields 7, 8, and 9.

Group Buffer Pool Activity Block in DB2 PM V5 Statistics

SYN.READS(XI)-DATA RETURNED **A**
SYN.READS(XI)-NO DATA RETURNED **B**
SYN.READS(NF)-DATA RETURNED **C**
SYN.READS(NF)-NO DATA RETURNED **D**

CLEAN PAGES SYNC.WRITTEN
CHANGED PAGES SYNC.WRITTEN
CLEAN PAGES ASYNC.WRITTEN
CHANGED PAGES ASYNC.WRITTEN

ASYNC.READS-DATA RETURNED **E**
ASYNC.READS-NO DATA RETURNED **F**

REG.PAGE LIST (RPL) REQUEST
CLEAN PAGES READ AFTER RPL **G**
CHANGED PGS READ AFTER RPL **H**

PAGES CASTOUT
CASTOUT CLASS THRESHOLD
GROUP BP CASTOUT THRESHOLD
CASTOUT ENGINE NOT AVAIL.

WRITE ENGINE NOT AVAILABLE
READ FAILED-NO STORAGE
WRITE FAILED-NO STORAGE

Note:

S in the formula to derive U for the group buffer pool size is the sum of fields **A**, **B**, **C**, and **D**.

A in the formula to derive U for the group buffer pool size is the sum of fields **E**, **F**, **G**, and **H**.

Example

100% data sharing
500 DASD writes per second, peak rate across Members 1 and 2
Group buffer pool page residency time of 120 sec
4 KB page size
Member 1 buffer pool configuration: VP=80,000 buffers, HP=160,000 buffers
Member 2 buffer pool configuration: VP=40,000 buffers, HP= 80,000 buffers

Data_entries $1 \times 500 \times 120 = 60{,}000$

Data(MB) $60{,}000 \times \frac{4}{1024} = 234$ MB

Directory_entries $60{,}000 + 1 \times (240{,}000 + 120{,}000) = 420{,}000$

Directory(MB) $1.1 \times 420{,}000 \times \frac{0.2}{1024} = 90$ MB

GBP(MB) 234 MB + 90 MB = 324 MB

RATIO $\frac{420{,}000}{60{,}000} = 7{:}1$

To provide you with a better overview of the variations, we present in Table 54 the variations resulting from a change in the data sharing percentage and a change in the residence time of updated pages in the group buffer pool.

Table 54. Group Buffer Pool Sizes and Directory-to-Data Ratios. These are for varying group buffer pool page residency times and data sharing percentages.

Average Page Residency Time in GBP (Seconds)	Data Sharing Percentage							
	10		20		50		100	
	(MB)	Ratio	(MB)	Ratio	(MB)	Ratio	(MB)	Ratio
30	14	25:1	28	25:1	70	25:1	139	25:1
60	20	13:1	40	13:1	100	13:1	201	13:1
90	26	9:1	53	9:1	131	9:1	263	9:1
120	32	7:1	65	7:1	162	7:1	324	7:1
180	45	5:1	90	5:1	224	5:1	448	5:1

For capacity planning, Table 54 enables you to choose the appropriate values and demonstrates the influence of page residency time and degree of data sharing on the GBP size and ratio.

5.13.4 Group Buffer Pool Assignment in a Real Case

As a study in diversity we show the results from a customer environment that had been studied in detail using three months of statistics gathered with a 30-min interval. Here the assumptions are 100% data sharing and 30 sec residence time of data in the group buffer pool before castout to DASD to become clean pages.

In Table 55 we present the calculation.

Table 55. Group Buffer Pool Assignments for a Real Case. This is taken out of a customer study.

Buffer Pool	Virtual Buffer Pool (MB)	Hiperpool (MB)	Maximum Pages Written to DASD per Minute	Group Buffer Pool (MB)	Directory-to-Data Ratio
BP0	20	40	88	7	18:1
BP10	30	240	4,681	206	2.5:1
BP11	40	400	1,673	91	7.6:1
BP20	40	80	265	17	12.3:1
BP21	40	240	76	17	94:1

Note: The sizes of the buffer pools are defined by their efficiency to save I/O rather than by hit rates. That is why virtual buffer pool and hiperpool sizes vary widely.

Buffer Pool Purpose

BP0	Catalog and directory
BP10	General data
BP11	General indexes
BP20	Big table spaces
BP21	Big table spaces indexes

Table 55 demonstrates the wide variation you will find in most DB2 environments. Here we separated large table spaces and their indexes out of the other table spaces, using as criterion size and access frequency. Out of a set of about 1000 table spaces, 6 qualified for virtual buffer pools BP20 and BP21. The indexes were always kept separate from the table space in a different virtual buffer pool.

Catalog and directory have their own buffer pools, as that is almost standard today. The same is true for the work file database, which is kept in its own virtual buffer pool 7. The update rate, here represented as the pages written per minute, varies by a factor of 50, and so does directory-to-data ratio. A high ratio reflects a more read-intensive environment where you have larger buffer pools and little write activity.

Chapter 6. Special Topics

In this chapter we discuss DB2 and RAMAC Virtual Array (RVA) storage and DB2 in a distributed environment.

6.1 DB2 and RAMAC Virtual Array Storage

In 2.7.1.2, “System Resources for DB2 Components” on page 23, we discuss DASD requirements for the DB2 system components and libraries and give the default values in terms of 3390 cylinders. However, in addition to the DB2 subsystem storage requirements, the size of the user databases can be anywhere from 50 GB to 1 TB or more, depending on the number of DB2 users, plans, application databases, and tables.

DASD space is therefore one of the critical system resources and as such any efficiency and performance improvements in it are translated directly into DB2 performance and throughput improvements.

IBM has a whole range of DASD offerings. An IBM Storage Products Specialist will be able to provide you with the detailed information about the offerings and help you design a suitable DASD solution for your DB2 applications. Because of the unique capabilities of the RVA and its excellent suitability for DB2 needs, we cover it in detail here.

6.1.1 IBM 9393 RAMAC Virtual Array

The IBM 9393 RVA is built on a virtual disk architecture and emulates both IBM 3380 and 3390 DASD. It delivers broad operational flexibility and a lower total cost of ownership while maintaining the high availability that is demanded by the mission-critical applications of your enterprise. The IBM 9393 RVA uses an advanced storage subsystem architecture where logical volumes are indirectly mapped to physical volumes within the subsystem. Unlike traditional storage subsystems, the IBM 9393 RVA maps user data across the subsystem. Operating in this manner, the IBM 9393 RVA can deliver outboard storage management and data reproduction functions. In addition, it can more effectively use storage capacity by storing only data that is actually used and not reserving capacity for unallocated space or space that has been allocated but is unused.

The latest models in the RVA series are the T42 and T82.

6.1.1.1 Features

The IBM 9393 RVA has the following general features:

- A compact, single-frame subsystem
- Outstanding environmental efficiency
- IBM-manufactured Ultrastar 2XP 4.5 GB disk drives
- Virtual disk architecture
- A comprehensive fault-tolerant design
- Built-in compression and compaction
- Support for 3380 and 3390 device types in single-, double-, and triple-volume capacities

- A self-tuning subsystem
- SnapShot support for rapid duplication (priced feature)
- Host software to manage configurations and monitor performance (separately priced product)
- Up to 726 GB effective capacity
- Up to 3072 MB of effective cache
- Up to 16 MB of effective nonvolatile storage

IBM RVA Model 9393-T42: RVA model 9393-T42 has the following features:

- Turbo shared memory for improved performance
- Up to four concurrent data transfers over ESCON channels
- 128 logical paths for improved ESCON connectivity
- Extended format data set support in MVS system-managed storage environments
- Upgrade capability from the base RVA model 002

IBM RVA Model 9393-T82: RVA model 9393-T82 has the following features:

- Turbo shared memory for improved performance
- Up to eight concurrent data transfers over ESCON channels
- 128 logical paths for improved ESCON connectivity
- Extended format data set support in MVS system-managed storage environments
- Upgrade capability from the base RVA model 002

6.1.1.2 Benefits for DB2

The log-structured design of the RVA has three unique advantages, as we discuss below.

SnapShot, a backup and copy product made possible by RVA′s log-structured design, can take essentially instantaneous point-in-time views of any desired volume or set of volumes (and of many supported types of data sets). The speed of taking copies is a major benefit of SnapShot. SnapShot is fast because you only copy the pointers, not the data. Thus SnapShot has the potential to remove hours from daily backup and copy operations.

The key feature of SnapShot, which distinguishes it from other duplication methods, is that *tracks do not have to be physically duplicated to establish the view*. Instead, two views of the data are represented in the form of two sets of track directory entries, both pointing to the same physical data. If either view is subsequently updated, the new tracks are written to the log in the standard manner called for by RVA′s log-structured design. Thus, duplicate tracks are created only to the extent that the two views of the data actually differ from each other.

The second advantage of the RVA′s log-structured design is virtual disks—a real benefit for DB2 because DB2 systems typically have a large allocated but unused disk requirement. Because there is no penalty for overallocation (that is, it does not occupy real space on the disk), you can define larger numbers of logical

volumes than the capacity you physically have installed. Page set expansion can be achieved easily, as the allocated and unused space becomes used, additional disk capacity can be added without having to reorganize the page sets. However, you are still required to have enough space available on your virtual (or “logical”) 3390 volumes.

In the RVA architecture, the definition of logical volumes is performed independently of the physical storage that supports such volumes. No logical-to-physical mapping occurs until the affected data is actually written, and the mapping evolves dynamically with time. Logical volumes of different types and sizes can be created at any time, as the need for them arises.

A key advantage of virtual disks is that physical storage can be *overcommitted.* By overcommitting physical storage, it is possible to achieve two simultaneous objectives:

- The host continues to use a logical volume of a given size (for example, 3390 model 3); therefore requests for added extents can be accommodated as easily as with real volumes of this size.
- Applications are spread across a larger number of logical addresses.

The first objective makes for a strategy that can be applied globally for a large pool of data, without needing to be concerned with causing extra B37 abends due to the use of small logical volumes. The advantage of the second objective is the reduction in the probability that two or more applications will both become active on the same address at the same time. Overcommitting the physical storage tends to improve both response time and throughput compared to a more conventional one-to-one relationship of logical and physical storage.

The third potentially important advantage of RVA′s log-structured design is its high throughput when performing random update writes. By taking advantage of its dynamic logical-to-physical mapping, the RVA groups all writes, including random ones, into large sequential batches. In this way, the RVA combines many destages in the same operation while also avoiding the RAID-5 “write penalty.” Thus, if a burst of random update writes occurs, the RVA can complete the destaging of the burst very rapidly. It handles burst writes better than the other subsystems because it has a higher back-end bandwidth to the disk than any of IBM′s other subsystems. Of course you have less cache than buffer pool, but usually the problem is destaging out of the cache quickly enough, which the RVA is good at.

In most workloads, random update writes tend to occur a few at a time and are easily handled within either the log-structured or RAID-5 framework. DB2 checkpoint, however, represents one specific case in which bursts of random update writes may be of special concern to some installations. The performance impact of DB2 checkpoint depends on the size of the DB2 buffer, the frequency of checkpoints, and the settings of the DB2 horizontal and vertical deferred write thresholds. If you have experienced performance problems associated with DB2 checkpoints, consider the RVA as an alternative, because of capability to provide high throughput for random update writes.

For further information about the latest RVA performance studies, contact your IBM Storage Product Specialist.

6.1.2 SnapShot

Although recommended for its unparalleled operational flexibility and very attractive cost of ownership, the RVA also offers several unique performance advantages, including SnapShot, virtual disk, and very fast destaging of random update write bursts. SnapShot in particular may provide many installations with a compelling reason to select RVA.

Many applications and situations require copying to be made of individual files or entire volumes. These applications include in-house backups, offsite backups, data mining, and testing. A very common situation is repeatable testing, when the test will manipulate the database.

Traditional facilities for making copies are expensive in both time and resources. With traditional copying facilities, it can take a long time to make a copy because data must be physically moved from one place to another. The data being copied is often unavailable to applications, to ensure it is in a consistent state. If the copy is to another disk, the capacity required to store the data is in addition to the capacity used to store the original file, even if the two copies are identical or (eventually) differ only slightly. And there is a performance cost on other work in the subsystem as data is physically copied from the source to the target location.

So, traditional copies are expensive in both time and resources.

SnapShot minimizes both time and resources. A SnapShot copy of data inside the RVA can be made in a few seconds regardless of the amount of data to be copied. There is no appreciable performance impact on other work running in the RVA. And the initial copy literally takes up no additional capacity at all. Over time, if the contents of either the original data or the copy change, additional capacity is used only to store the changes.

All RVA models now offer SnapShot enablement at the data set, volume, and minidisk levels. SnapShot allows you to “rethink” how and when data can be duplicated because it does not use the media, channel, CPU cycle, and memory resources required by traditional data copy methods. This capability can provide dramatic savings of time and money while enabling you to improve quality and productivity in such applications as year-2000 conversions, data mining, and backup.

Note

SnapShot is really IBM RAMAC SnapShot (for MVS or VM/ESA), and it is a program product that also requires a feature on the RVA.

SnapShot has no impact on the subsystem when taking a snap, apart from the fact that you direct a few I/Os to the target and source volumes, but those are minimal. There is an overhead if you subsequently take copies to tape, but that is the same as without SnapShot. With SnapShot you can more easily defer backups to spread the dump load over time.

SnapShot is not perfect for DB2, but if you are taking point-in-time copies of data, either using DFSMSdss or DB2 utilities, SnapShot can make the copies more quickly. If you are a high-availablility 24x7 installation, you probably do not want to quiesce DB2 to take a SnapShot. If you are using SHRLEVEL(CHANGE), SnapShot does not really buy anything for image copy.

At the moment using SnapShot for image copy requires RECOVER LOGONLY, and you basically take the copy outside DB2. IBM has issued a statement of direction about enhancing the concurrent copy interface to allow the DB2 COPY utility to automatically invoke SnapShot. This enhancement is subject to the same restrictions as concurrent copy is today but significantly enhances usability.

Here is the statement of direction for DFSMS/MVS SnapShot support:

As an expanding number of IBM customers install RAMAC Virtual Array Storage and begin exploiting IBM RAMAC SnapShot for MVS/ESA, enablement of SnapShot from DFSMS/MVS would greatly enhance its usability. IBM intends to enhance the Concurrent Copy interface of the DFSMSdss component of DFSMS/MVS to exploit SnapShot. Programs using the Concurrent Copy interface, such as CICS, DB2 and IMS, will be able to take advantage of SnapShot capability.

6.2 DB2 in a Distributed Environment

In this section we present the DB2 workstation products that support distributed environment. We also list some performance and planning conclusions and recommendations based on some of the DB2 client/server performance studies conducted by IBM.

6.2.1 DB2 Products for Distributed DB2 Support

DB2 Universal Database Version 5 (DB2 UDB), announced in December 1996, is the follow-on product to DB2 common servers.

DB2 UDB supports server operation in AIX, OS/2, Windows NT, Windows 95, HP-UX, and Solaris operating environments. DB2 UDB server products support database clients for all server platforms as well as Macintosh, Siemens-Nixdorf (SINIX), SCO OpenServer, Silicon Graphics IRIX (SGI), and Windows 3.x. DB2 UDB supports single-user operation on OS/2, Windows NT, and Windows 95.

DB2 Connect is the follow-on product to Distributed Database Connection Services (DDCS). DB2 Connect allows stand-alone users or local area network (LAN) clients to access data stored in an application server such as DB2 for

OS/390, DB2 for OS/400, and DB2 for VM and VSE, using distributed relational database architecture (DRDA). Applications running on the client machines work with the host data transparently, as if a local database server managed the data.

DB2 Connect supports IBM system network architecture (SNA) through advanced program-to-program communication (APPC) and advanced peer-to-peer networking (APPN) for connections with DRDA servers. It also supports transmission control protocol/internet control (TCP/IP) for connections with DRDA servers. For the current releases, however, only DB2 for OS/390 and UDB products support the TCP/IP protocol.

Figure 26 illustrates a simple distributed DB2 setup configuration, although not all combinations are currently supported.

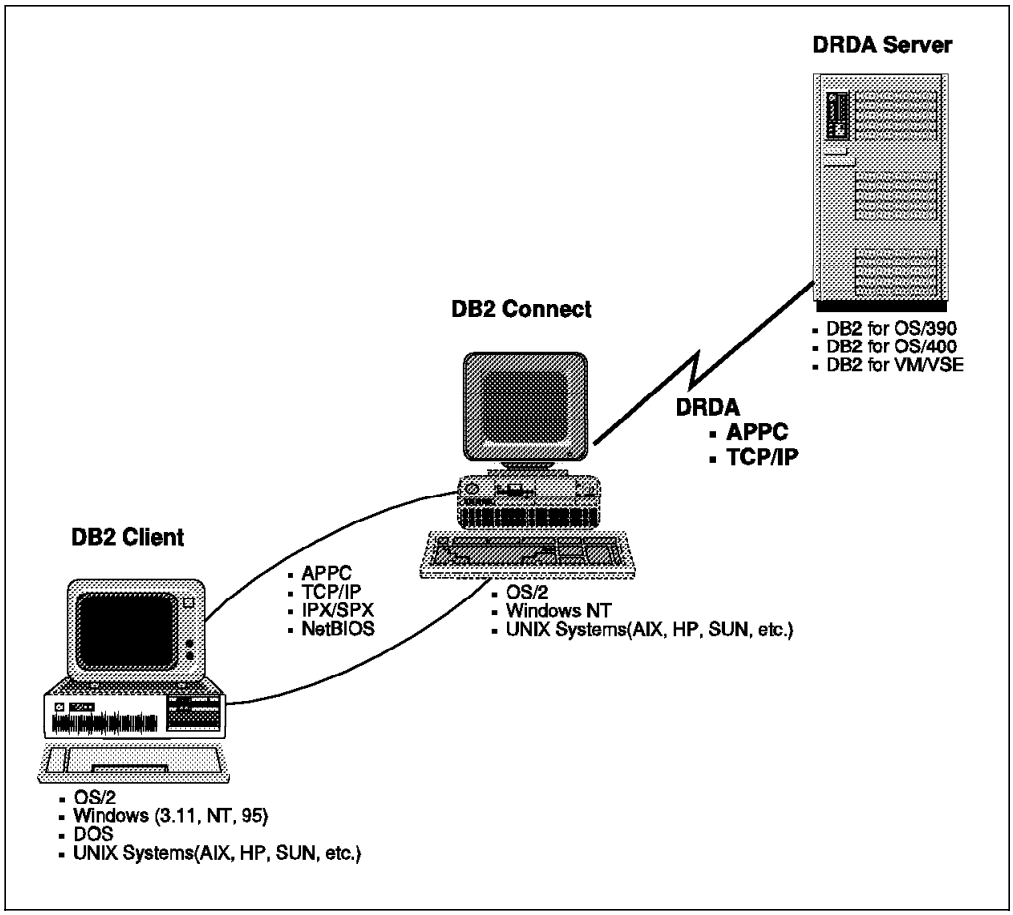


Figure 26. Simple Scenario with DB2 Connect

DB2 Connect runs on different platforms, including Windows 3.11, Windows 95, Windows NT, OS/2, AIX, HP, Sun, SINIX, and SCO. The Windows 3.11 and Windows 95 products are available only in single-user versions.

DRDA uses the logical unit 6.2 (LU6.2) SNA support for:

- DDCS
- DB2 common server
- DB2 for MVS/ESA up to Version 4
- DB2 for OS/390 version 5

6.2.2 Performance and Planning Considerations for Distributed DB2

The number of factors influencing the performance of distributed DB2 are numerous and depend heavily on the numerous components of client systems, common servers, DRDA server, the communication protocols, and the system and software resources assigned to each client and server.

Throughput and performance measurement studies have been conducted at IBM with certain sets of configurations representing different client/server scenarios. Because every client/server setup is unique in setup and requirements, no measurement numbers can be directly applied to a particular distributed DB2 environment. However, from the results and analysis of a fair number of performance measurements for different client/server relational database configurations, certain conclusions were reached. These conclusions helped us produce a set of considerations to guide you in planning a DB2 client/server configuration and having realistic performance expectations of it. We list some considerations below. For a full discussion of these studies, have your DB2 product specialist provide you with the *DB2 Family:Client/Server Performance Measurement Series* white papers related to your environment. These papers are also available on the Internet at **http://www.software.ibm.com/data/db2/performance** under *DB2 Family: Client/Server Performance.*

General Considerations—DRDA Server

- When using stored procedures in a client/server environment, tune the database as if the access is local.
- Although the COMMIT_ON_RETURN available in DB2 V5 does not commit when the stored procedure ends, it forces a commit when control is returned to the client, avoiding unnecessary locks due to bad programming.
- Avoid overutilizing the CPU that runs DDCS or DB2 Connect if the clients are holding locks on the target database. Overutilizing the CPU delays freeing of the locks, which can affect client access to the same tables from other sources.
- Buffer pool allocation has an effect on CPU utilization and throughput. Make sure the buffer pool is not set to a value that causes paging to occur when additional clients are added or when some other system task occurs.

General Considerations—DB2 Client

- The more processors available on the hardware, the greater the capacity for DB2 Connect.
- DB2 Connect should run on a fast processor. A two-processor machine is significantly faster than a one-processor machine of the same processor speed. A four-processor machine is marginally faster than a two-processor machine. Consider upgrading from two to four processors if capacity is the goal, not speed.
- DB2 Connect should run with enough memory to handle the number of expected clients. Plan for 200 KB for each client passing through DB2 Connect for NT and OS/2. The base level of RAM should be based on the number of applications and RAM requirements for those applications. We suggest that DB2 Connect be the only application running on the machine.
- Hard disk capacity is not an issue for DB2 Connect be the process does not require large amounts of disk capacity.

Network Considerations: The network plays a big part in how well the client/server application performs, and how great an effect client/server applications have on database performance. Message transmission delay is the biggest factor, as the locks are held until the message moves from the client to the database. For most measurements, stored procedures were used to decrease the number of SQL calls so as to reduce the effect of locks. The 3172 mod 1 and LAN were used for the measurements. For the mix of transactions, the message lengths were large, so the following values were set:

- 4 KB RUSIZE in VTAM
- 32 KB RQRIOBLK definition in DB2 Connect for NT
- 32 KB RQRIOBLK definition in the OS/2 clients
- 32 KB RQRIOBLK definitions in the Win95 clients

Network Control Program (NCP) Considerations: The NCP runs in IBM communication controllers like the 3745. The DELAY parameters in the NCP play an important role in the performance of the NCP as well as the response time observed at the clients. Here are some of the general considerations for the NCP setup in a client/server environment:

- Assess the line speeds between the LAN and DB2 for OS/390. The throughput is only as good as the slowest line. Overutilizing any line results in delays.
- Introducing multiple communication devices (for example, 3745s, bridges, routers) on the communication path between the LAN and the database introduces delays.
- Tune the NCP definition DELAY parameters PCCU and GROUP LNCNTL=CA. If response time at the client is most important, set all DELAY values to zero. If the communication control unit (CCU) is constrained, set the GROUP LNCNTL=CA to 0.1 or 0.2; 0.2 creates a greater response time delay than 0.1.

LAN Considerations: Most tests were done in one very specific environment where there was very light LAN utilization. Typically, a production environment has more activity on the LAN. Query environments place a large burden on the network (LAN and wide area network (WAN)). Monitor your LAN to make sure it is not being overutilized. This is especially true for transactions that return large answer sets. Overutilization of the LAN could introduce delays with resultant bad effects on database performance.

WAN Considerations: The WAN portion of the network is generally where the majority of time is spent. Performance can be affected by the type of devices as well as how these devices are configured. The following are some typical connectivity solutions:

- 3745

  LAN attached (TIC) provides medium to high throughput, depending on the TIC type and 3745 model. 3745s are also used for connections over long distance by connecting between two 3745s. The line speed of the connection between the 3745s should be as fast as possible to provide the best response time.

- OSA

  The mainframe is directly connected to the LAN, which provides very high throughput.

- 3172

  This can provide LAN/channel attach capabilities as well as “3172 to 3172” attach capabilities for long distance. The LAN/channel attach provides very high throughput. The 3172 to 3172 configuration throughput depends on the line speed connecting the two.

- ESCON

  This is a channel attached solution where DB2 Connect is directly channel attached to the mainframe. It does not require DB2 Connect traffic to traverse the LAN as an OSA, 3172, or 3745 solution requires. This solution provides very high throughput with multipath channel (MPC) support providing the best throughput.

The configuration parameters for these devices, VTAM, and TCP/IP have large to small effects on response time, dependent on the parameter. Listed below are some of these parameters. They are not all valid in all configurations.

- DELAY

  Most devices have one or more DELAY parameters that can be set. This parameter has severe performance implications because the network message can be held up enroute to and from the client. This has a direct effect on response time, so it should be set to zero in all cases. Watch out for defaults, as the default values tend to be nonzero. For 3745 connectivity, two definitions contain DELAY parameters: the PCCU and the LNCTL=CA. The PCCU is most important. Note also that in some configurations, the PCCU macro is replaced by a PU definition.

- MAXOUT (VTAM PU)

  Set MAXOUT to 7.

- PASSLIM (VTAM PU)

  Set PASSLIM to 7.

- MAXDATA (VTAM definition)

  Set this value large enough to contain the largest PIU. If set to 33100, it would be large enough to handle RUs up to 32 KB.

- VPACING (VTAM APPL)

  Set VPACING to 16.

- MAXTSL (NCP LINE)

  If connecting through a 3745 TIC, set MAXTSL to 16372.

- MAXBFRU (VTAM definition)

  Make sure this is large enough to avoid slowdown conditions.

6.2.2.1 Further Reading and Planning Material for Distributed DB2

Communications Server for OS/2 Warp information,
http://www.raleigh.ibm.com/cm2/cm2prod.html

IBM Software Servers information,
http://www.software.ibm.com/is/sw-servers

VTAM V4R4 Network Implementation Guide, SC31-8370

VTAM V4R4 Resource Definition Reference, SC31-8377

VTAM Programming for LU 6.2 Reference, SC31-8375

TCP/IP Performance and Tuning Guide, SC31-7188-02

TCP/IP V3R2 for MVS: Programmer's Reference, SC31-7135-02

DRDA Connectivity Guide, SC26-4783-03

WOW! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database, SG24-2212.

DB2 UDB Building Applications for UNIX, S10J-8161

DB2 UDB Administration, Getting Started, S10J-8154

Planning for Integrated Networks, SC31-7123

Appendix A. CP90 M-Values

Table 56 lists the CP90 M-values for IBM processors for the OS/390 operating system (MVS/ESA). The table is current as of August 1997.

Table 56 (Page 1 of 2). OS/390 M-Values from the CP90 Default M-Values Table (August 1997)

Machine	Engines	CP90 Default M-Value	Machine	Engines	CP90 Default M-Value	Machine	Engines	CP90 Default M-Value
9672-RY5	10	19328	9672-RX5	10	17309	9672-R95	9	16521
9672-R85	8	15567	9672-R75	7	14384	9672-R65	6	13001
9672-R55	5	11439	9672-R45	4	8997	9672-R35	3	7100
9672-RC5	3	6289	9672-R25	2	5033	9672-RB5	2	3818
9672-R15	1	2710	9672-RA5	1	2129	9672-RY4	10	16054
9672-RX4	10	14595	9672-R94	9	13854	9672-R84	8	12907
9672-R74	7	11780	9672-R64	6	10489	9672-R54	5	9048
9672-R44	4	7474	9672-R34	3	5798	9672-RC4	3	5185
9672-R24	2	4010	9672-RB4	2	2741	9672-R14	1	2096
9672-RA4	1	1448	2003-156	5	6064	2003-146	4	5314
2003-136	3	4182	2003-1C5	3	3821	2003-135	3	3344
2003-126	2	3002	2003-125	2	2394	2003-124	2	2219
2003-116	1	1626	2003-115	1	1323	2003-107	1	1078
2003-106	1	714	2003-105	1	555	2003-104	1	395
2003-103	1	237	2003-102	1	158	9672-RX3	10	7442
9672-R83	8	6425	9672-R73	7	5820	9672-R63	6	5153
9672-R53	5	4398	9672-R72	7	4703	9672-R52	5	3773
9672-R42	4	3177	9672-R32	3	2499	9672-R22	2	1750
9672-R12	1	936	9672-RA2	1	658	9672-R61	6	2811
9672-R51	5	2509	9672-R41	4	2144	9672-R31	3	1710
9672-R21	2	1213	9672-R11	1	658	9021-9X2	10	20807
9021-982	8	17417	9021-972	7	15651	9021-962	6	13801
9021-952	5	11810	9021-942	4	9625	9021-941	4	9267
9021-832	3	7397	9021-831	3	7177	9021-822	2	5087
9021-821	2	4966	9021-711	1	2696	9021-900	6	10526
9021-860	5	9046	9021-820	4	7383	9021-740	3	5406
9021-720	6	5222	9021-660	2	3912	9021-640	2	3774
9021-620	4	3649	9021-580	3	2857	9021-520	1	2058
9021-500	2	1960	9021-340	1	1047	9021-330	1	946
9121-742	4	4683	9121-732	3	3661	9121-622	2	2541
9121-621	2	2542	9121-522	2	2032	9121-521	2	2065
9121-511	1	1362	9121-411	1	1078	9121-311	1	886
9121-610	4	3010	9121-570	3	2338	9121-490	2	1624
9121-480	2	1619	9121-440	2	1255	9121-320	1	881

Table 56 (Page 2 of 2). OS/390 M-Values from the CP90 Default M-Values Table (August 1997)

Machine	Engines	CP90 Default M-Value	Machine	Engines	CP90 Default M-Value	Machine	Engines	CP90 Default M-Value
9121-260	1	672	9121-210	1	465	9121-190	1	314
9121-180	1	214	9221-421	2	1265	9221-221	2	780
9221-211	1	700	9221-201	1	580	9221-191	1	468
9221-200	2	441	9221-170	1	264	9221-150	1	202
9221-130	1	139	9221-120	1	78	3090-28T	2	1920
3090-25T	2	1201	3090-18T	1	1047	3090-17T	1	827
3090-15T	1	664	3090-600J	6	5222	3090-600S	6	4406
3090-600E	6	3262	3090-500J	5	4456	3090-500S	5	3794
3090-500E	5	2946	3090-400J	4	3649	3090-400S	4	3134
3090-400E	4	2442	3090-380J	3	2796	3090-380S	3	2428
3090-300J	3	2857	3090-300S	3	2539	3090-300E	3	1974
3090-280J	2	1920	3090-280S	2	1681	3090-280E	2	1317
3090-250J	2	1038	3090-250S	2	952	3090-200J	2	1960
3090-200S	2	1750	3090-200E	2	1372	3090-180J	1	1047
3090-180S	1	946	3090-180E	1	744	3090-170J	1	744
3090-170S	1	659	3090-150J	1	582	3090-150S	1	528
3090-150E	1	457	3090-120J	1	404	3090-120S	1	339
3090-120E	1	339	3090-110J	1	339	3090-100S	1	230
4381-92E	2	379	4381-91E	1	201	4381-90E	1	160

Appendix B. DB2 PM Sample Statistics Reports

This appendix gives DB2 PM reports that are referenced in the body of the book.

The DB2 PM reports contained herein are taken from *DATABASE 2 for MVS/ESA Version 4 Data Sharing Performance Topics* (SG24-4611) and refer to the IRWW.

B.1 DB2 PM Statistics Report: Buffer Pool Detail

```
   LOCATION: DSNDBOG                        DB2 PERFORMANCE MONITOR (V4)                                PAGE: 1-16
      GROUP: DSNDBOG                          STATISTICS REPORT - LONG                  REQUESTED FROM: 09/09/95 08:41:15.00
     MEMBER: DB1G                                                                                        TO: 09/09/95 08:51:17.00
  SUBSYSTEM: DB1G                                                                      INTERVAL FROM: 09/09/95 08:41:16.36
DB2 VERSION: V4                                    SCOPE: MEMBER                                         TO: 09/09/95 08:51:16.73

---- HIGHLIGHTS ------------------------------------------------------------------------------------------------------
INTERVAL START  : 09/09/95 08:41:16.36    SAMPLING START: 09/09/95 08:41:16.36    TOTAL THREADS      :     0.00
INTERVAL END    : 09/09/95 08:51:16.73    SAMPLING END  : 09/09/95 08:51:16.73    TOTAL COMMITS      : 18056.00
INTERVAL ELAPSED:         10:00.377445    OUTAGE ELAPSED:            0.000000    DATA SHARING MEMBER:      N/A

BP5    GENERAL                QUANTITY  /MINUTE  /THREAD  /COMMIT    BP5    READ OPERATIONS        QUANTITY  /MINUTE  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------    ---------------------------  --------  -------  -------  -------
CURRENT ACTIVE BUFFERS          46.00      N/A      N/A      N/A    GETPAGE REQUEST              15157.00  1514.75      N/C     0.84
UNAVAIL.BUFFER-VPOOL FULL        0.00     0.00      N/C     0.00    GETPAGE REQUEST-SEQUENTIAL    4473.00   447.02      N/C     0.25
                                                                   GETPAGE REQUEST-RANDOM       10684.00  1067.73      N/C     0.59
NUMBER OF DATASET OPENS          0.00     0.00      N/C     0.00
                                                                   SYNCHRONOUS READS             4310.00   430.73      N/C     0.24
BUFFERS ALLOCATED - VPOOL     6250.00   624.61      N/C     0.35    SYNCHRON. READS-SEQUENTIAL    4178.00   417.54      N/C     0.23
BUFFERS ALLOCATED - HPOOL        0.00     0.00      N/C     0.00    SYNCHRON. READS-RANDOM         132.00    13.19      N/C     0.01
HPOOL BUFFERS BACKED             0.00     0.00      N/C     0.00
                                                                   GETPAGE PER SYN.READ-RANDOM     80.94
DFHSM MIGRATED DATASET           0.00     0.00      N/C     0.00
DFHSM RECALL TIMEOUTS            0.00     0.00      N/C     0.00    SEQUENTIAL PREFETCH REQUEST   8554.00   854.86      N/C     0.47
                                                                   SEQUENTIAL PREFETCH READS        0.00     0.00      N/C     0.00
HPOOL EXPANS. OR CONTRACT.       0.00     0.00      N/C     0.00    PAGES READ VIA SEQ.PREFETCH      0.00     0.00      N/C     0.00
VPOOL EXPANS. OR CONTRACT.       0.00     0.00      N/C     0.00    S.PRF.PAGES READ/S.PRF.READ       N/C
VPOOL OR HPOOL EXP.FAILURE       0.00     0.00      N/C     0.00
                                                                   LIST PREFETCH REQUESTS           0.00     0.00      N/C     0.00
CONCUR.PREF.I/O STREAMS-HWM      0.00      N/A      N/A      N/A    LIST PREFETCH READS              0.00     0.00      N/C     0.00
PREF.I/O STREAMS REDUCTION       0.00     0.00      N/C     0.00    PAGES READ VIA LIST PREFTCH      0.00     0.00      N/C     0.00
PARALLEL QUERY REQUESTS          0.00     0.00      N/C     0.00    L.PRF.PAGES READ/L.PRF.READ       N/C
PARALL.QUERY REQ.REDUCTION       0.00     0.00      N/C     0.00
PREF.QUANT.REDUCED TO 1/2        0.00     0.00      N/C     0.00    DYNAMIC PREFETCH REQUESTED       0.00     0.00      N/C     0.00
PREF.QUANT.REDUCED TO 1/4        0.00     0.00      N/C     0.00    DYNAMIC PREFETCH READS           0.00     0.00      N/C     0.00
                                                                   PAGES READ VIA DYN.PREFETCH      0.00     0.00      N/C     0.00
BUFFERPOOL EXPANSIONS             N/A      N/A      N/A      N/A    D.PRF.PAGES READ/D.PRF.READ       N/C

                                                                   PREF.DISABLED-NO BUFFER       8554.00   854.86      N/C     0.47
                                                                   PREF.DISABLED-NO READ ENG        0.00     0.00      N/C     0.00

                                                                   SYNC.HPOOL READ                  0.00     0.00      N/C     0.00
                                                                   ASYNC.HPOOL READ                 0.00     0.00      N/C     0.00
                                                                   HPOOL READ FAILED                0.00     0.00      N/C     0.00
                                                                   ASYN.DA.MOVER HPOOL READ-S       0.00     0.00      N/C     0.00
                                                                   ASYN.DA.MOVER HPOOL READ-F       0.00     0.00      N/C     0.00

                                                                   PAGE-INS REQUIRED FOR READ       0.00     0.00      N/C     0.00
```

Figure 27. DB2 PM Statistics Report: Buffer Pool Detail

B.2 DB2 PM Statistics Report: Locking Activity

```
   LOCATION: DSNDBOG                        DB2 PERFORMANCE MONITOR (V4)                          PAGE: 1-4
      GROUP: DSNDBOG                          STATISTICS REPORT - LONG                REQUESTED FROM: 09/09/95 08:41:15.00
     MEMBER: DB1G                                                                                 TO: 09/09/95 08:51:17.00
  SUBSYSTEM: DB1G                                                                     INTERVAL FROM: 09/09/95 08:41:16.36
DB2 VERSION: V4                                    SCOPE: MEMBER                                  TO: 09/09/95 08:51:16.73

---- HIGHLIGHTS ------------------------------------------------------------------------------------------------------
INTERVAL START  : 09/09/95 08:41:16.36    SAMPLING START: 09/09/95 08:41:16.36    TOTAL THREADS      :     0.00
INTERVAL END    : 09/09/95 08:51:16.73    SAMPLING END  : 09/09/95 08:51:16.73    TOTAL COMMITS      : 18056.00
INTERVAL ELAPSED:         10:00.377445    OUTAGE ELAPSED:            0.000000    DATA SHARING MEMBER:      N/A

LOCKING ACTIVITY             QUANTITY  /MINUTE  /THREAD  /COMMIT    DATA SHARING LOCKING         QUANTITY  /MINUTE  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------    --------------------------   --------  -------  -------  -------
SUSPENSIONS (ALL)             3391.00   338.89      N/C     0.19    LOCK REQUESTS (P-LOCKS)      25890.00  2587.37      N/C     1.43
SUSPENSIONS (LOCK ONLY)        343.00    34.28      N/C     0.02    UNLOCK REQUESTS (P-LOCKS)    25855.00  2583.87      N/C     1.43
SUSPENSIONS (LATCH ONLY)      3048.00   304.61      N/C     0.17    CHANGE REQUESTS (P-LOCKS)       10.00     1.00      N/C     0.00
SUSPENSIONS (OTHER)              0.00     0.00      N/C     0.00
                                                                   SYNCH.XES - LOCK REQUESTS      128.9K    12.9K      N/C     7.14
TIMEOUTS                         0.00     0.00      N/C     0.00    SYNCH.XES - CHANGE REQUESTS  64952.00  6491.12      N/C     3.60
DEADLOCKS                        0.00     0.00      N/C     0.00    SYNCH.XES - UNLOCK REQUESTS    119.6K    12.0K      N/C     6.62
                                                                   ASYNCH.XES - RESOURCES           0.00     0.00      N/C     0.00
LOCK REQUESTS                  147.5K    14.7K      N/C     8.17
UNLOCK REQUESTS              54317.00  5428.29      N/C     3.01    SUSPENDS - IRLM GLOBAL CONT    621.00    62.06      N/C     0.03
QUERY REQUESTS                   0.00     0.00      N/C     0.00    SUSPENDS - XES GLOBAL CONT.      0.00     0.00      N/C     0.00
CHANGE REQUESTS              65204.00  6516.30      N/C     3.61    SUSPENDS - FALSE CONTENTION   1016.00   101.54      N/C     0.06
OTHER REQUESTS                   0.00     0.00      N/C     0.00    INCOMPATIBLE RETAINED LOCK       0.00     0.00      N/C     0.00

LOCK ESCALATION (SHARED)         0.00     0.00      N/C     0.00    NOTIFY MESSAGES SENT           692.00    69.16      N/C     0.04
LOCK ESCALATION (EXCLUSIVE)      0.00     0.00      N/C     0.00    NOTIFY MESSAGES RECEIVED         0.00     0.00      N/C     0.00
                                                                   P-LOCK/NOTIFY EXITS ENGINES     10.00      N/A      N/A      N/A
DRAIN REQUESTS                   0.00     0.00      N/C     0.00    P-LCK/NFY EX.ENGINE UNAVAIL      0.00     0.00      N/C     0.00
DRAIN REQUESTS FAILED            0.00     0.00      N/C     0.00
CLAIM REQUESTS                 165.8K    16.6K      N/C     9.18    PSET/PART P-LCK NEGOTIATION      0.00     0.00      N/C     0.00
CLAIM REQUESTS FAILED            0.00     0.00      N/C     0.00    PAGE P-LOCK NEGOTIATION        388.00    38.78      N/C     0.02
                                                                   OTHER P-LOCK NEGOTIATION         0.00     0.00      N/C     0.00
                                                                   P-LOCK CHANGE DURING NEG.      388.00    38.78      N/C     0.02

GLOBAL DDF ACTIVITY          QUANTITY  /MINUTE  /THREAD  /COMMIT    QUERY PARALLELISM            QUANTITY  /MINUTE  /THREAD  /COMMIT
---------------------------  --------  -------  -------  -------    --------------------------   --------  -------  -------  -------
DBAT QUEUED-MAXIMUM ACTIVE        N/P      N/P      N/P      N/A    MAX.DEGREE OF PARALLELISM        0.00      N/A      N/A      N/A
CONV.DEALLOC-MAX.CONNECTED        N/P      N/P      N/P      N/A    PARALLEL GROUPS EXECUTED         0.00     0.00      N/C     0.00
INACTIVE DBATS - CURRENTLY        N/P      N/A      N/A      N/A    EXECUTED AS PLANNED              0.00     0.00      N/C     0.00
INACTIVE DBATS - HWM              N/P      N/A      N/A      N/A    REDUCED DEGREE - NO BUFFER       0.00     0.00      N/C     0.00
COLD START CONNECTIONS            N/P      N/P      N/P      N/P    FALL TO SEQUENTIAL-CURSOR        0.00     0.00      N/C     0.00
WARM START CONNECTIONS            N/P      N/P      N/P      N/P    FALL TO SEQUENTIAL-NO ESA        0.00     0.00      N/C     0.00
RESYNCHRONIZATION ATTEMPTED       N/P      N/P      N/P      N/P    FALL TO SEQUENTIAL-NO BUFF.      0.00     0.00      N/C     0.00
RESYNCHRONIZATION SUCCEEDED       N/P      N/P      N/P      N/P    FALL TO SEQUENTIAL-ENCL.SER      0.00     0.00      N/C     0.00
```

Figure 28. DB2 PM Statistics Report: Locking Activity

B.3 DB2 PM Statistics Report: SQL Activity

```
   LOCATION: DSNDBOG                        DB2 PERFORMANCE MONITOR (V4)                              PAGE: 1-1
      GROUP: DSNDBOG                          STATISTICS REPORT - LONG                  REQUESTED FROM: 09/09/95 08:41:15.00
     MEMBER: DB1G                                                                                     TO: 09/09/95 08:51:17.00
  SUBSYSTEM: DB1G                                                                      INTERVAL FROM: 09/09/95 08:41:16.36
DB2 VERSION: V4                                    SCOPE: MEMBER                                      TO: 09/09/95 08:51:16.73

---- HIGHLIGHTS ------------------------------------------------------------------------------------------------------
INTERVAL START  : 09/09/95 08:41:16.36    SAMPLING START: 09/09/95 08:41:16.36    TOTAL THREADS      :     0.00
INTERVAL END    : 09/09/95 08:51:16.73    SAMPLING END  : 09/09/95 08:51:16.73    TOTAL COMMITS      : 18056.00
INTERVAL ELAPSED:         10:00.377445    OUTAGE ELAPSED:            0.000000    DATA SHARING MEMBER:      N/A

SQL DML                     QUANTITY  /MINUTE  /THREAD  /COMMIT    SQL DCL                     QUANTITY  /MINUTE  /THREAD  /COMMIT
--------------------------  --------  -------  -------  -------    --------------------------  --------  -------  -------  -------
SELECT                      86981.00  8692.63      N/C     4.82    LOCK TABLE                      0.00     0.00      N/C     0.00
INSERT                      53011.00  5297.77      N/C     2.94    GRANT                           0.00     0.00      N/C     0.00
UPDATE                      64530.00  6448.94      N/C     3.57    REVOKE                          0.00     0.00      N/C     0.00
DELETE                       2505.00   250.34      N/C     0.14    SET HOST VARIABLE               0.00     0.00      N/C     0.00
                                                                   SET CURRENT SQLID               0.00     0.00      N/C     0.00
PREPARE                         0.00     0.00      N/C     0.00    SET CURRENT DEGREE              0.00     0.00      N/C     0.00
DESCRIBE                        0.00     0.00      N/C     0.00    SET CURRENT RULES               0.00     0.00      N/C     0.00
DESCRIBE TABLE                  0.00     0.00      N/C     0.00
OPEN                        81417.00  8136.58      N/C     4.51    CONNECT TYPE 1                  0.00     0.00      N/C     0.00
CLOSE                       45883.00  4585.42      N/C     2.54    CONNECT TYPE 2                  0.00     0.00      N/C     0.00
FETCH                         166.8K    16.7K      N/C     9.24    RELEASE                         0.00     0.00      N/C     0.00
                                                                   SET CONNECTION                  0.00     0.00      N/C     0.00
TOTAL                         501.2K    50.1K      N/C    27.76
                                                                   TOTAL                           0.00     0.00      N/C     0.00

SQL DDL                     QUANTITY  /MINUTE  /THREAD  /COMMIT    STORED PROCEDURES           QUANTITY  /MINUTE  /THREAD  /COMMIT
--------------------------  --------  -------  -------  -------    --------------------------  --------  -------  -------  -------
CREATE TABLE                    0.00     0.00      N/C     0.00    CALL STATEMENTS EXECUTED        0.00     0.00      N/C     0.00
CREATE INDEX                    0.00     0.00      N/C     0.00    PROCEDURE ABENDS                0.00     0.00      N/C     0.00
CREATE VIEW                     0.00     0.00      N/C     0.00    CALL STATEMENT TIMEOUTS         0.00     0.00      N/C     0.00
CREATE SYNONYM                  0.00     0.00      N/C     0.00    CALL STATEMENT REJECTED         0.00     0.00      N/C     0.00
CREATE TABLESPACE               0.00     0.00      N/C     0.00
CREATE DATABASE                 0.00     0.00      N/C     0.00
CREATE STOGROUP                 0.00     0.00      N/C     0.00
CREATE ALIAS                    0.00     0.00      N/C     0.00

ALTER  TABLE                    0.00     0.00      N/C     0.00
ALTER  INDEX                    0.00     0.00      N/C     0.00
ALTER  TABLESPACE               0.00     0.00      N/C     0.00
ALTER  DATABASE                 0.00     0.00      N/C     0.00
ALTER  STOGROUP                 0.00     0.00      N/C     0.00

DROP   TABLE                    0.00     0.00      N/C     0.00
DROP   INDEX                    0.00     0.00      N/C     0.00
DROP   VIEW                     0.00     0.00      N/C     0.00
DROP   SYNONYM                  0.00     0.00      N/C     0.00
DROP   TABLESPACE               0.00     0.00      N/C     0.00
DROP   DATABASE                 0.00     0.00      N/C     0.00
DROP   STOGROUP                 0.00     0.00      N/C     0.00
DROP   ALIAS                    0.00     0.00      N/C     0.00
DROP   PACKAGE                  0.00     0.00      N/C     0.00

COMMENT ON                      0.00     0.00      N/C     0.00
LABEL ON                        0.00     0.00      N/C     0.00

TOTAL                           0.00     0.00      N/C     0.00
```

Figure 29. DB2 PM Statistics Report: SQL Activity

B.4 DB2 PM Statistics Report: Group Buffer Pool Totals

```
   LOCATION: DSNDBOG                        DB2 PERFORMANCE MONITOR (V4)                          PAGE: 1-30
      GROUP: DSNDBOG                          STATISTICS REPORT - LONG                 REQUESTED FROM: 09/09/95 08:41:15.00
     MEMBER: DB1G                                                                                  TO: 09/09/95 08:51:17.00
  SUBSYSTEM: DB1G                                                                      INTERVAL FROM: 09/09/95 08:41:16.36
DB2 VERSION: V4                                    SCOPE: MEMBER                                   TO: 09/09/95 08:51:16.73

---- HIGHLIGHTS ------------------------------------------------------------------------------------------------------
INTERVAL START  : 09/09/95 08:41:16.36    SAMPLING START: 09/09/95 08:41:16.36    TOTAL THREADS      :     0.00
INTERVAL END    : 09/09/95 08:51:16.73    SAMPLING END  : 09/09/95 08:51:16.73    TOTAL COMMITS      : 18056.00
INTERVAL ELAPSED:         10:00.377445    OUTAGE ELAPSED:             0.000000    DATA SHARING MEMBER:      N/A

GROUP TOT4K                 QUANTITY  /MINUTE  /THREAD  /COMMIT
--------------------------  --------  -------  -------  -------
SYN.READS(XI)-DATA RETURNED  22279.00  2226.50      N/C     1.23
SYN.READS(XI)-R/W INTEREST      10.00     1.00      N/C     0.00
SYN.READS(XI)-NO R/W INTER.      0.00     0.00      N/C     0.00
SYN.READS(NF)-DATA RETURNED  27537.00  2751.97      N/C     1.53
SYN.READS(NF)-R/W INTEREST   81112.00  8106.10      N/C     4.49
SYN.READS(NF)-NO R/W INTER.      0.00     0.00      N/C     0.00
ASYNC.READS-DATA RETURNED        0.00     0.00      N/C     0.00
ASYNC.READS-R/W INTEREST         0.00     0.00      N/C     0.00
ASYNC.READS-NO R/W INTEREST      0.00     0.00      N/C     0.00

CLEAN PAGES SYNC.WRITTEN         0.00     0.00      N/C     0.00
CHANGED PAGES SYNC.WRITTEN     110.7K    11.1K      N/C     6.13
CLEAN PAGES ASYNC.WRITTEN        0.00     0.00      N/C     0.00
CHANGED PAGES ASYNC.WRITTEN     56.00     5.60      N/C     0.00

PAGES CASTOUT                61901.00  6186.21      N/C     3.43
CASTOUT CLASS THRESHOLD          0.00     0.00      N/C     0.00
GROUP BP CASTOUT THRESHOLD      64.00     6.40      N/C     0.00
CASTOUT ENGINE NOT AVAIL.        0.00     0.00      N/C     0.00

WRITE ENGINE NOT AVAILABLE       0.00     0.00      N/C     0.00
READ FAILED-NO STORAGE           0.00     0.00      N/C     0.00
WRITE FAILED-NO STORAGE          0.00     0.00      N/C     0.00

OTHER REQUESTS               28564.00  2854.60      N/C     1.58

STATISTICS REPORT COMPLETE
```

Figure 30. DB2 PM Statistics Report: Group Buffer Pool Totals

Appendix C. Buffer Pool Tuning

In this appendix we describe a simple approach to measuring the efficiency of your buffer pools. It is based on some work by Mike Barnard of GA Life, York, UK, that he presented at a DB2 Guide meeting on June 5, 1997 at IBM South Bank, London. The original presentation can be extracted from the Website of the UK DB2 working group of the Guide Share Europe organization at http://www.gseukdb2.org.uk/.

The methodology is based on the observation that one of the objectives of the buffer pool is to avoid DASD read I/Os by holding pages in memory in the expectation that they will be accessed again in a short period of time. We first define a "reread" as the second or subsequent read of a page during a fixed period of time. A reread occurs when one page frame is stolen for use by a different page between the first and second GETPAGE. The buffer pool manager steals pages when there are no other unused pages, on a least recently used (LRU) basis.

Rather than use the buffer pool hit ratio, which is a relative measure, you can now define the reread percentage (Figure 31).

$$\text{Reread percentage} = \frac{\text{Number of reread I/Os}}{\text{Number of read I/Os}} \times 100$$

where the I/Os are synchronous random reads

Figure 31. Reread Percentage Formula

The original paper presented two methods for measuring the number of rereads, the first being statistically correct but complex to measure, the second giving a slightly lower result for the reread percentage but being much simpler to calculate.

We present here the second method:

1. Use DB2 PM or SMF to capture the trace data.

   From DB2 PM or directly in DB2 outputting to SMF, execute the following trace for a fixed period of time.

   ```
   -START TRACE(ACCTG) CLASS(30) IFCID(6)
   ```

   You can do this either from DB2 PM or directly in DB2 outputting to SMF. We suggest you start with a small interval, such as 1 min, so that you can gauge any effect on system performance and get a measure of the quantity of trace data being generated at your site. You can constrain the trace by both AUTHID and PLAN if the volume of data is too large.

   The original research arrived at a value of 10 min for the trace period by plotting a graph of the number of rereads against the interval between each. The majority of rereads occurred within a few minutes but with a reasonable tail extending out to 10 min. We suggest a period of between 5 and 10 min depending on the profile of your business transactions.

2. Use DB2 PM to format the trace data into a report.

   Execute the DB2 PM batch job with the following commands:

```
RECTRACE
     TRACE
        FROM   (,HH:MM:SS)
        TO     (,HH:MM:SS)
        LEVEL  (SHORT)
        INCLUDE(IFCID(6))
EXEC
```

Define a data set for the report output:

```
//RTTRCDD1 DD  DSN=userid.DB2PM.IFCID6.REPORT,
//             DISP=(NEW,CATLG),UNIT=3390,SPACE=(CYL,(nn,n),RLSE)
```

This will produce a report like that shown in Figure 32. Note that we have left in the printing carriage control characters.

```
1   LOCATION: DB41                          DB2 PERFORMANCE MONITOR (V4)                          PAGE: 1-1
      GROUP: N/P                                RECORD TRACE - SHORT                      REQUESTED FROM: NOT SPECIFIED
     MEMBER: N/P                                                                                    TO: NOT SPECIFIED
  SUBSYSTEM: DB41                                                                        ACTUAL FROM: 09/22/97 17:10:42.21
DB2 VERSION: V4                                                                            PAGE DATE: 09/22/97
OPRIMAUTH CONNECT
 ORIGAUTH CORRNAME  CONNTYPE      RECORD TIME     DEST  ACE IFC  DESCRIPTION                        DATA
 PLANNAME CORRNMBR  INSTANCE      TCB CPU TIME   SEQ NO     ID
 -------- -------- ----------- ----------------- ------ --- --- -------------- ------------------------------------------------------
 HUN17    TSO                 17:10:42.21145525      2   1   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
 HUN17    HUN17    TSO               0.26131100                 START         DBID    6         POOL ID          0  ACE            1
 DSNESPCS 'BLANK'  AF4D30BE0764                                                OBID    9         FIRST    X'0001E1'  READTYPE       R
                                    17:10:42.24696075      3   1   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
                                          0.26304200                 START         DBID    6         POOL ID          0  ACE            1
                                                                                   OBID    92        FIRST    X'000004'  READTYPE       R
                                    17:10:42.27707137      4   1   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
                                          0.26488200                 START         DBID    6         POOL ID          0  ACE            1
                                                                                   OBID    9         FIRST    X'000641'  READTYPE       R
                                    17:10:42.30205312      5   1   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
                                          0.27473150                 START         DBID    6         POOL ID          0  ACE            1
                                                                                   OBID    9         FIRST    X'00063F'  READTYPE       R
                                    17:10:42.36893750      6   1   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
                                          0.31903550                 START         DBID    269       POOL ID          3  ACE            1
                                                                                   OBID    88        FIRST    X'000002'  READTYPE       R
                                    17:11:22.43840875    289   2   6 READ I/O   --> NETWORKID:  'BLANK'   LUNAME:  'BLANK'   LUWSEQ:     1
                                          1.61683850                 START         DBID    6         POOL ID          0  ACE            1
                                                                                   OBID    9         FIRST    X'000E00'  READTYPE       S
```

Figure 32. DB2 PM Record Trace: IFCID 6

This example is for DB2 PM Version 4. In Version 5 the FIRST data field increases from 6 bytes to 8 to allow for LARGE table spaces, and an extra line is added per trace event, field name TABLE_SPACE_TYPE, with L for a LARGE table space and N for a non-LARGE table space. The values of READTYPE are:

S Sequential prefetch request

L List prefetch request

D Dynamic sequential prefetch request

R Synchronous read request

Note that for the prefetch operations, only the first page number is captured, and therefore it is unlikely to give a true measure of the number of rereads. To find the number of actual pages read in a prefetch operation, trace IFCID 7.

3. Create a DB2 table to hold the report data.

   Create a DB2 table, using the following SQL:

```
CREATE TABLE IFCID6
       ( POOL_ID       CHAR(8)       NOT NULL
       , DBID          SMALLINT      NOT NULL
       , OBID          SMALLINT      NOT NULL
       , FIRST         CHAR(6)       NOT NULL
       , READTYPE      CHAR(1)       NOT NULL
       )
```

Note that we have assumed that the DBID and the OBID have not been translated from the internal number to the object name. If this has happened in your report, define DBID and OBID as CHAR(8) and adjust the load SYSIN parameters accordingly. This table does not capture the start time of the read I/O and therefore cannot be used to calculate the interval time between the initial read and a reread. The time is printed on the report on the first line of the block of three output lines per IFCID 6 event. You could capture the time as well by using the ISPF editor to copy the ACE characters over column position 118 through 120 and then amending the LOAD utility control statements accordingly.

4. Execute the LOAD utility to load the data into the table.

   Use the following LOAD utility statements to load the data from the report into the DB2 table:

```
LOAD DATA INDDN SYSREC00 REPLACE LOG NO CONTINUEIF(118:120)='ACE'
 INTO TABLE HUONA.IFCID6 WHEN (80:83)='DBID'
 ( POOL_ID    POSITION(108:115)  CHAR(8)
 , DBID       POSITION(88:91)    INTEGER EXTERNAL
 , OBID       POSITION(218:221)  INTEGER EXTERNAL
 , FIRST      POSITION(239:244)  CHAR(6)
 , READTYPE   POSITION(260:260)  CHAR(1)
         )
```

   Note that the continuation field of length 3 bytes is removed from every input record before any concatenation takes place. This is why the READTYPE column is picked up from position 260, that is, $266 - (2 \times 3) = 260$

5. Execute the following SQL query to calculate the reread percentage for each buffer pool:

```
SELECT POOL_ID
      , SUM(READ) AS READS
      , SUM(REREAD) AS REREADS
      , 100*SUM(REREAD)/SUM(READ) AS REREAD_PERCENTAGE
FROM (SELECT POOL_ID
           , DBID
           , OBID
           , FIRST
           , COUNT(*) AS READ
           , COUNT(*) - 1 AS REREAD
      FROM HUONA.IFCID6
      WHERE READTYPE = 'R'
      GROUP BY POOL_ID
             , DBID
             , OBID
             , FIRST
      ) AS T
GROUP BY POOL_ID
```

   If you are using DB2 V3, define a view for the nested table expression. The output from this query if you use SPUFI will look like this:

```
---------+---------+---------+---------+---------+---------+------
POOL_ID   READTYPE        READS      REREADS  REREAD_PERCENTAGE
---------+---------+---------+---------+---------+---------+------
       0  R                5166          310                  6
       1  R               12828         1796                 14
       2  R               54755         9856                 18
```

6. Resize the buffer pools according to the results and repeat from step 1. In the original work, a target of less than 10% was set for the reread percentage for random synchronous I/Os (READTYPE=R).

Appendix D. Special Notices

This publication is intended to help DB2 professionals in their capacity planning activities. The information in this publication is not intended as the specification of any programming interfaces that are provided by DB2 for OS/390 Version 5. See the PUBLICATIONS section of the IBM Programming Announcement for DB2 for OS/390 Version 5 for more information about what publications are considered to be product documentation.

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

CICS	CICS/ESA
CICS/MVS	DATABASE 2
DB2	DFSMS
DFSMS/MVS	DFSMSdss
DFSMShsm	DFSORT
DRDA	IBM
IMS	IMS/ESA
MVS/ESA	OS/390
RACF	RAMAC
RMF	S/390
SNAP/SHOT	Sysplex Timer
System/390	3090

The following terms are trademarks of other companies:

C-bus is a trademark of Corollary, Inc.

Java and HotJava are trademarks of Sun Microsystems, Incorporated.

Microsoft, Windows, Windows NT, and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

Pentium, MMX, ProShare, LANDesk, and ActionMedia are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

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

Appendix E. Related Publications

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

E.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see “How To Get ITSO Redbooks” on page 151.

- *Selecting a Server - The Value of S/390*, SG24-4812
- *OS/390 MVS Parallel Sysplex Capacity Planning*, SG24-4680
- *OS/390 MVS Parallel Sysplex Configuration Cookbook*, SG24-4706
- *DB2 for OS/390, Application Design Guidelines for High Performance*, SG24-2233
- *DB2 for OS/390 Version 5 Performance Topics*, SG24-2213
- *DB2 for MVS/ESA Version 4 Non-Data-Sharing Performance Topics,* SG24-4562
- *DB2 for MVS/ESA Version 4 Data Sharing Performance Topics*, SG24-4611
- *DB2 V3 Performance Topics*, GG24-4284
- *DB2 PM Usage Guide Update*, SG24-2584
- *Planning for Conversion to the DB2 Family: Methodology and Practice*, GG24-4445
- *WOW! DRDA Supports TCP/IP: DB2 Server for OS/390 and DB2 Universal Database*, SG24-2212
- *IBM RAMAC Virtual Array*, SG24-4951
- *Disaster Recovery Library: S/390 Technology Guide*, SG24-4210
- *Disaster Recovery Design Concepts*, SG24-4211
- *Disaster Recovery Library: Data Recovery*, GG24-3994
- *DDCS/2 to DB2 Performance Benchmarks*, GG24-4308

E.2 Redbooks on CD-ROMs

Redbooks are also available on CD-ROMs. **Order a subscription** and receive updates 2-4 times a year at significant savings.

CD-ROM Title	Subscription Number	Collection Kit Number
System/390 Redbooks Collection	SBOF-7201	SK2T-2177
Networking and Systems Management Redbooks Collection	SBOF-7370	SK2T-6022
Transaction Processing and Data Management Redbook	SBOF-7240	SK2T-8038
AS/400 Redbooks Collection	SBOF-7270	SK2T-2849
RS/6000 Redbooks Collection (HTML, BkMgr)	SBOF-7230	SK2T-8040
RS/6000 Redbooks Collection (PostScript)	SBOF-7205	SK2T-8041
Application Development Redbooks Collection	SBOF-7290	SK2T-8037
Personal Systems Redbooks Collection	SBOF-7250	SK2T-8042

E.3 Other Publications

These publications are also relevant as further information sources:

- *DB2 for OS/390 Version 5 Data Sharing: Planning and Administration*, SC26-8961
- *DB2 for OS/390 Version 5 Administration Guide*, SC26-8957
- *DB2 for OS/390 Version 5 Installation Guide*, GC26-8970
- *DB2 for MVS/ESA Version 4 Administration Guide*, SC26-3265
- *DB2 Version 3 Administration Guide*, SC26-4888
- *DB2 PM for OS/390 Version 5 Online Monitor User's Guide*, SC26-8990
- *DB2 PM for OS/390 Version 5 Batch User's Guide*, SC26-8991
- *DB2 PM for OS/390 Version 5 Report Reference, Volume 1*, SC26-8985
- *DB2 PM for OS/390 Version 5 Report Reference, Volume 2*, SC26-8986
- *OS/390 MVS Programming, Sysplex Services Reference*, GC28-1772-02
- *OS/390 MVS Setting Up a Sysplex*, GC28-1779-02
- *System/390 MVS Sysplex Overview: An Introduction to Data Sharing and Parallelism*, GC28-1208
- *System 390 PR/SM Planning Guide*, GA22-7236-01
- *VTAM V4R4 Network Implementation Guide*, SC31-8370
- *VTAM V4R4 Resource Definition Reference*, SC31-8377
- *VTAM Programming for LU 6.2 Reference*, SC31-8375
- *DRDA Connectivity Guide*, SC26-4783-03
- *TCP/IP Performance and Tuning Guide*, SC31-7188-02
- *TCP/IP V3R2 for MVS: Programmer's Reference*, SC31-7135-02
- *Planning for Integrated Networks*, SC31-7123
- *DB2 UDB Building Applications for UNIX*, S10J-8161
- *DB2 UDB Administration, Getting Started*, S10J-8154

How To Get ITSO Redbooks

This section explains how both customers and IBM employees can find out about ITSO redbooks, CD-ROMs, workshops, and residencies. A form for ordering books and CD-ROMs is also provided.

This information was current at the time of publication, but is continually subject to change. The latest information may be found at URL http://www.redbooks.ibm.com/redbooks.

How IBM Employees Can Get ITSO Redbooks

Employees may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **PUBORDER** — to order hardcopies in United States
- **GOPHER link to the Internet** - type GOPHER.WTSCPOK.ITSO.IBM.COM
- **Tools disks**

  To get LIST3820s of redbooks, type one of the following commands:

  ```
  TOOLS SENDTO EHONE4 TOOLS2 REDPRINT GET SG24xxxx PACKAGE
  TOOLS SENDTO CANVM2 TOOLS REDPRINT GET SG24xxxx PACKAGE (Canadian users only)
  ```

  To get lists of redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET ITSOCAT TXT
  TOOLS SENDTO USDIST MKTTOOLS MKTTOOLS GET LISTSERV PACKAGE
  ```

  To register for information on workshops, residencies, and redbooks:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1996
  ```

  For a list of product area specialists in the ITSO:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Home Page on the World Wide Web**

  http://w3.itso.ibm.com/redbooks/redbooks.html

- **IBM Direct Publications Catalog on the World Wide Web**

  http://www.elink.ibmlink.ibm.com/pbl/pbl

  IBM employees may obtain LIST3820s of redbooks from this page.

- **ITSO4USA category on INEWS**
- **Online** — send orders to: USIB6FPL at IBMMAIL or DKIBMBSH at IBMMAIL
- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank). A category form and detailed instructions will be sent to you.

How Customers Can Get ITSO Redbooks

Customers may request ITSO deliverables (redbooks, BookManager BOOKs, and CD-ROMs) and information about redbooks, workshops, and residencies in the following ways:

- **Online Orders** (Do not send credit card information over the Internet) — send orders to:

	IBMMAIL	**Internet**
In United States:	usib6fpl at ibmmail	usib6fpl@ibmmail.com
In Canada:	caibmbkz at ibmmail	lmannix@vnet.ibm.com
Outside North America:	bookshop at dkibmbsh at ibmmail	bookshop@dk.ibm.com

- **Telephone orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	(long distance charges apply)
(+45) 4810-1320 - Danish	(+45) 4810-1020 - German
(+45) 4810-1420 - Dutch	(+45) 4810-1620 - Italian
(+45) 4810-1540 - English	(+45) 4810-1270 - Norwegian
(+45) 4810-1670 - Finnish	(+45) 4810-1120 - Spanish
(+45) 4810-1220 - French	(+45) 4810-1170 - Swedish

- **Mail Orders** — send orders to:

IBM Publications	IBM Publications	IBM Direct Services
Publications Customer Support	144-4th Avenue, S.W.	Sortemosevej 21
P.O. Box 29554	Calgary, Alberta T2P 3N5	DK-3450 Allerød
Raleigh, NC 27626-0570	Canada	Denmark
USA		

- **Fax** — send orders to:

United States (toll free)	1-800-445-9269
Canada (toll free)	1-800-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1) 415 855 43 29 (Outside USA)** — ask for:

  Index # 4421 Abstracts of new redbooks
  Index # 4422 IBM redbooks
  Index # 4420 Redbooks for last six months

- **Direct Services** - send note to softwareshop@vnet.ibm.com
- **On the World Wide Web**

Redbooks Home Page	http://www.redbooks.ibm.com/redbooks
IBM Direct Publications Catalog	http://www.elink.ibmlink.ibm.com/pbl/pbl

- **Internet Listserver**

  With an Internet E-mail address, anyone can subscribe to an IBM Announcement Listserver. To initiate the service, send an E-mail note to announce@webster.ibmlink.ibm.com with the keyword subscribe in the body of the note (leave the subject line blank).

IBM Redbook Order Form

Please send me the following:

Title	Order Number	Quantity

- **Please put me on the mailing list for updated versions of the IBM Redbook Catalog.**

First name Last name

Company

Address

City Postal code Country

Telephone number Telefax number VAT number

- Invoice to customer number _____
- Credit card number _____

Credit card expiration date Card issued to Signature

We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.

DO NOT SEND CREDIT CARD INFORMATION OVER THE INTERNET.

Index

E

G

I

L

M

O

P

Q

R

S

T

V

ITSO Redbook Evaluation

DB2 for OS/390 Capacity Planning
SG24-2244-00

Your feedback is very important to help us maintain the quality of ITSO redbooks. **Please complete this questionnaire and return it using one of the following methods:**

- Use the online evaluation form found at http://www.redbooks.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redeval@vnet.ibm.com

Please rate your overall satisfaction with this book using the scale:
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction _____

Please answer the following questions:

Was this redbook published in time for your needs? Yes____ No____

If no, please explain:

_____

_____

_____

_____

What other redbooks would you like to see published?

_____

_____

_____

Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

_____

_____

_____

_____

_____

Printed in U.S.A.