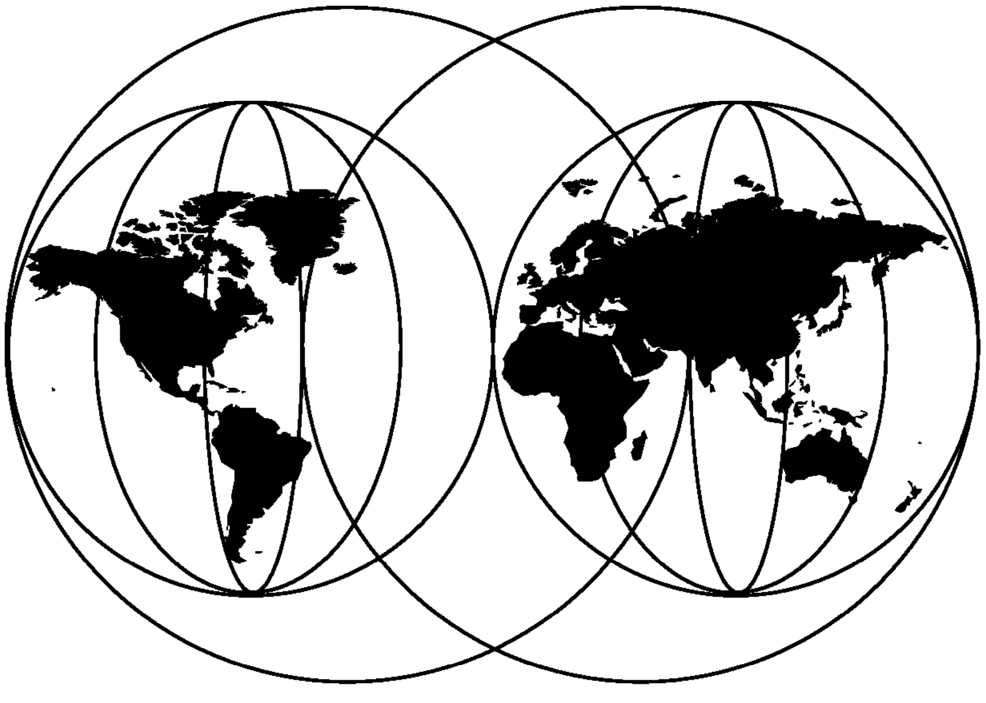
Benchmarking in Focus

Andy Hoetzel, Alain Benhaim, Nigel Griffiths, Chet Holliday
Norbert Pistoor



International Technical Support Organization

http://www.redbooks.ibm.com

SG24-5052-00

SG24-5052-00

International Technical Support Organization

Benchmarking in Focus

March 1998

Take Note!

Before using this information and the product it supports, be sure to read the general information in Appendix E, “Special Notices” on page 161.

First Edition (March 1998)

This book applies to projects on non-standard customer-defined benchmarks. Some technical sections refer to IBM RS/6000 and the AIX Operating System.

Comments may be addressed to:
IBM Corporation, International Technical Support Organization
Dept. JN9B Building 045 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

Contents

Figures

Tables

Preface

This redbook will help you successfully complete performance-related measurements or tests (commonly called benchmarks) by giving a detailed description of the various stages that make up a benchmark project, such as defining requirements and objectives, planning activities, preparing resources, executing tests, and reporting results.

Extensive checklists are provided to help benchmark participants keep track of the progress of the benchmark project. The Appendixes also contain references to other sources of benchmarking information.

This redbook is intended to help project leaders, project managers, and consultants plan a benchmark project. It will also help technical specialists, technical marketing representatives and system engineers who are involved in the execution of benchmark tests and the presentation of their results. Sales representatives will find the less-technical chapters useful in assessing the efforts and costs related to a proposed benchmark project.

Since the authors of this redbook have a strong RS/6000 background, most of the examples come from the RS/6000 environment. Nevertheless, we believe that the principles underlying a successful benchmark apply to all environments.

To benefit from this redbook, the reader should have some basic understanding of the technology of modern information systems, including hardware and software. Detailed knowledge of specific hardware or software products is not assumed.

How This Book Is Organized

Chapter 1 contains an introduction into the subject of benchmarking and explains our approach to benchmark projects.

Chapter 2 reviews the basic concepts used in benchmarking, which are essential to understanding of the following chapters.

Chapter 3 provides details to consider before beginning a benchmark project, and discusses some alternatives. This should help with the decision of whether to do a benchmark or not.

Chapter 4 provides an overview of the process that should be followed in a successful benchmark project.

Chapters 5 to 10 contain detailed information about the different phases into which a benchmark project can be divided. This covers definition, planning, preparation, execution, reporting, and follow-up tasks.

The Appendixes contain useful information on how to get support from IBM RS/6000 benchmark centers, as well as checklists that help the participants in a benchmarking project keep track of the various activities in which they are involved.

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 Austin Center.

Andy Hoetzel is an International Technical Support Specialist for RS/6000 and AIX Performance at the International Technical Support Organization, Austin Center. He writes extensively and teaches IBM classes worldwide on all areas of AIX internals, performance and tuning. Andy holds a master of science degree in computer science from the University of Texas at El Paso. Before joining the ITSO, Andy worked in the AIX Competence Center in Munich, Germany as an AIX Technical Support Specialist.

Alain Benhaim is a parallel database Specialist in France specialized in Oracle and DB2. He has six years of experience in AIX. He is presently working in the IBM Parallel Solutions Support Center in Montpellier where he is mostly involved in commercial benchmarks on the RS/6000 SP.

Nigel Griffiths is a Performance Guru in U.K. He has 18 years of experience with UNIX, five of which are with IBM. His areas of expertise include C programming from system tools to UNIX kernel internals and performance tuning and sizing of large SMP and parallel SP Oracle databases. He had been the U.K. Benchmark Team Leader for four years.

Chet Holliday is a Senior Marketing Support Representative in the RS/6000 Benchmarking Center in Dallas, Texas. He is ″Team Lead″ of the staff that conducts RS/6000 benchmark projects. The team also handles Business Recovery System activities and handling leasing of equipment. Chet has 31 years experience with IBM, has been leading benchmark projects for the last ten years, and RS/6000 benchmarks since the product′s initial release.

Norbert Pistoor is a Consultant at the IBM Midrange Systems Sales Technical Support Center in Munich, Germany. He has more than eight years of experience with RS/6000, including three years in benchmarking. He holds a PhD in physics from University of Mainz, Germany. His areas of expertise include benchmarking, performance analysis, high availability, and computer simulations in physics.

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

Marcus Brewer
International Technical Support Organization, Austin Center, Texas, USA

John Weiss
International Technical Support Organization, Austin Center, Texas, USA

Tara Campbell
International Technical Support Organization, Austin Center, Texas, USA

Steve Gardner
International Technical Support Organization, Austin Center, Texas, USA

Jasenn McNair
International Technical Support Organization, Austin Center, Texas, USA

Terry Rosser
International Technical Support Organization, Austin Center, Texas, USA

Janice Hawley
International Technical Support Organization, Austin Center, Texas, USA

Pokil Wong
RS/6000 Product Specialist, IBM in Los Angeles, California, USA

Vance Sutton
RS/6000 SP Customer Benchmark Center, IBM in Poughkeepsie, New York, USA

Francois Thomas
RS/6000 SP Scientific Benchmark Specialist, IBM in Montpellier, France

Chris Eisenmann
Advisory Marketing Support Representative, RS/6000 Benchmark Center, Dallas, Texas, USA

Carol Borges
Administrative Support, RS/6000 Benchmark Center, Dallas, Texas, USA

Heidemarie Hoetzel
Software Engineer, IBM, Germany

Comments Welcome

Your comments are important to us!

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

- Fax the evaluation form found in "ITSO Redbook Evaluation" on page 173 to the fax number shown on the form.
- Use the electronic evaluation form found on the Redbooks Web sites:

For Internet users	`http://www.redbooks.ibm.com`
For IBM Intranet users	`http://w3.itso.ibm.com`

- Send us a note at the following address:

  `redbook@vnet.ibm.com`

Chapter 1. Introduction

Within the information technology (IT) community, the term benchmark is used quite extensively, and with varying definitions.

Generically, a *benchmark* is a set of conditions that are used as a reference. This definition is widely used in various fields, not only information technology, but also in economics or finance. Fund managers in the international stock markets, for example, use indices (averages of prices of a selection of stocks considered to represent the respective market) as a benchmark to compare the performance of their portfolios to the overall market performance.

In information technology, there exist various benchmarks in the sense of this definition, among them the popular *SPECint95* and *SPECfp95* which were established by the Standard Performance Evaluation Corporation (SPEC), a non-profit corporation whose members are more than 40 companies, many of them vendors of workstations and servers. These benchmarks are averages of runtimes of various small programs which are considered to represent various kinds of applications.

Another set of standard benchmarks, *TPC-C* and *TPC-D*, comes from the Transaction Processing Performance Council (TPC). Whereas the SPEC benchmarks are meant to be a set of general purpose benchmarks, the TPC benchmarks deal specifically with transaction oriented applications using a database. They measure response times (how long it takes to get the result of a query) and throughput (how many queries can be run during a given time) of a given set of queries against a given set of data in a database.

Since these standardized benchmarks usually can give only a first estimate of the expected behavior of a given application running on a particular system, more and more decision makers do not rely simply on standard benchmark results. They want proof that the system they are going to buy really can handle the task it will be assigned to. Therefore, they want to run their own benchmark that would be designed specifically to meet their requirements.

Many of them, however, are not aware of the efforts and costs associated with this procedure. This has led to the habit of calling almost every kind of test involving some sort of measurement related to performance issues a benchmark. But, as we will show in this redbook, there are many aspects to consider and many pitfalls to avoid in order to get meaningful results from a benchmarking project. And in the end it turns out that those benchmarks that stick quite closely to the original meaning of the word usually are the most rewarding ones.

1.1 Divide and Conquer

Like many other simple questions, the question ″Which system should I buy for my application?″ quickly gets more and more complex the more one thinks about it. We therefore suggest a structured approach which divides the task into several pieces and tackles them separately.

1.1.1 Need to Know

As in any other field, when benchmark specialists talk to each other they tend to use special vocabulary to indicate quickly what they are thinking about. Anybody who wants to enter the field of benchmarking should know of the basic concepts behind the terms in use, such as

- Classification of benchmarks
- Benchmark environment and workload
- Methods of benchmarking

All you need to know about these and related topics can be found in Chapter 2, “Benchmark Basics” on page 7.

1.1.2 Assessing the Necessity

Before entering into a probably complex and expensive benchmark project, it could be worthwhile to consider some alternatives. Do I really need a benchmark? In order to answer this question we need to collect some basic information:

- What is the proposed environment?
- What tasks will it be assigned to?
- What do I know about the workload, the amount and structure of data to be stored, transferred, or manipulated?
- Are there installations with similar characteristics that could be used as a reference?
- Can I do some back-of-the-envelope calculations which give me a sense of how reasonable the proposed environment is?
- Will the result of a benchmark have a significant influence on the decision of which system will be bought, or are there other criteria that seem to be more important to the decision makers?

These, and similar questions, as well as how to answer these questions will be discussed in Chapter 3, “To Benchmark or Not to Benchmark” on page 25.

1.1.3 How the Process Works

Once we have decided that we want to do a benchmark, it is time to become familiar with the process of managing a benchmark project. We suggest you divide the project into several phases, which will then be expanded in more detail later in this book. These are

- Definition
- Planning
- Preparation
- Execution
- Reporting

Chapter 4, “Benchmark Process Overview” on page 39 gives an overview of what should be the basic milestones in the project.

1.1.4 Defining Objectives and Requirements

As a first step, we need to be more specific about what we must know in order to have some facts that help to select the right environment. Questions to answer include:

- Who are the different parties participating in the project? What are their interests? What will be their share of the execution of the project?
- What will be the configuration of the system to be tested?
- What will be measured? How will it be measured?
- What is the workload to be used on the system?
- What are the criteria for success?

Implications of these and other questions, as well as solutions, and how to handle them will be discussed in Chapter 5, “The Definition Phase” on page 47.

1.1.5 Making a Plan

The next step is to make a detailed plan of the different activities needed and identify the people who are responsible.

- How do we perform the measurements we have defined?
- How should the environment be implemented (such as database layout)?
- How should the workload be created?
- What resources are needed in order to do the measurements (hardware, software, skills)?
- Where are the critical points in our plan? What risks are involved?
- How can we prepare for unexpected problems (for example, data loss)?

- How much time and money are required for the activities?
- Do all parties agree to this benchmark plan?

More details about this and related topics can be found in Chapter 6, "The Planning Phase" on page 55.

1.1.6 Setting the Stage

After the plan is complete and agreed upon, but before the actual benchmark can be done, several things have to be arranged:

- Installing hardware and software
- Customizing the basics of the system
- Checking some basic functions (for example, network connections)
- Optimizing the participants' working environment

More about these activities will be provided in Chapter 7, "The Preparation Phase" on page 65.

1.1.7 Executing the Tasks

Here we are at the heart of the matter. The customer has arrived at the site of the benchmark and we will be

- Installing the benchmark code
- Loading the data and integrating the workload
- Ensuring system operation
- Generating terminal emulation scripts
- Determining problems and solving them
- Running the tests
- Analyzing system performance and tuning the system
- Collecting the results

All this and more hints and tips relating to this part of the project are contained in Chapter 8, "The Execution Phase" on page 73.

1.1.8 Reporting Results

Now that the measurements have been done and the results look reasonable we need to analyze them to interpret them correctly.

- Is the meaning of the numbers fully understood?
- Were we able to meet our success criteria?
- What are the reasons for deviations and what are their consequences?
- How should we present our findings to the interested parties?

Chapter 9, "The Reporting Phase" on page 93 will deal with various aspects of how to interpret benchmark results and how to present them.

1.1.9 Success or Defeat

Some time after we finish our job we will be interested in knowing whether or not we were successful. Maybe we can also assist with clearing some last doubts or explain again some of our findings. We should also get some feedback from the other participants in the effort on what they think about the project.

Please refer to Chapter 10, "The Follow-Up Phase" on page 103 for more about this.

1.2 How to Get Help

If you do not have your own benchmark center, you might want to get some help from people who support such projects regularly (including the authors of this book). Please refer to Appendix A, "IBM RS/6000 Benchmark Centers" on page 109 for some more information about the services the IBM RS/6000 Benchmark Centers can provide. Since this might change from time to time, do not forget to check the relevant web sites, too.

Similar institutions exist for other brands and from other vendors as well.

1.3 How to Keep Track

Since we will be involved in many different tasks we need to keep track of the progress our project makes. For the success of the project, it is essential that the participants have all the information they need for their particular tasks, as well as for the project as a whole.

Appendix B, "Checklists and Sample Forms" on page 113, contains checklists and sample forms which can be used as a starting point. They can easily be adapted to the needs of your particular project.

1.4 Useful Tools

There are many tools that can be used during the various stages of a benchmark project. Appendix C, "Tools Used in Benchmarking" on page 151 gives a short description of the tools the authors find particularly useful when taking part in benchmarks on IBM RS/6000 servers and workstations.

More detail about the tools can be found in their respective documentation.

1.5 Industry Standard Benchmarks

While this book is primarily about commercial non-standard customer-defined benchmarks, we feel it would be incomplete without mentioning industry standard benchmarks and their fields of application.

Appendix D, “Industry Standard Benchmarks” on page 157, describes some of the most popular industry standard benchmarks and provides sources for more detailed information on that subject.

Chapter 2. Benchmark Basics

In this chapter we will explain the most important ingredients for a successful benchmark project. This will provide you with the basic understanding of terms and concepts that will be used in the following chapters.

2.1 Reasons for Benchmarking

As we will see in this section, there are various reasons for proposing a benchmark project. Understanding the purpose of the project is the most important factor to ensure success. For many decisions that we have to make during the project, remembering what the purpose of the project is will set the direction in which we have to proceed.

The purpose of a benchmark project is also used to classify the benchmark as belonging to one of several different types. Table 1 on page 8 contains the most common ones.

Table 1. Purpose of the Benchmark

Benchmark Type	Purpose
Competitive **	To obtain the best result possible using equipment that is to be sold for a price in a given range. The vendor with the best result is supposed to win.
Hurdle	To demonstrate that at least a particular level of performance can be reached. Typically used by customers making a short list of vendors.
Sizing	To make sure the recommended machine is the right configuration by proving that certain performance related requirements are met.
System or Soak Test	To test the application on a machine that is more powerful than the development system. This usually requires a fully-configured benchmark machine.
Proof of Concept or Working Demo	To show that the recommended components of the solution will actually work together and nothing has been forgotten.
Vendor Test	To demonstrate that customer and vendor can work together as partners on a complex project. This type might come disguised as one of the other types. Performance numbers need not be an important outcome of the project.
** **Note**: Competitive benchmarks are sometimes called shoot outs or head-to-head benchmarks against the competition which highlights the competition for the sale.	

2.2 Conditions of Benchmarking

When thinking about a benchmark project, there are many requirements stated in the form of conditions that will have to be met. These conditions will provide the basis of what will be the *environment* the benchmark will be executed in and what will be the *workload* that is imposed upon this environment.

Obtaining precise formulations of what these conditions are will be the first step in the benchmarking project. In this section we will give some general ideas of what these conditions might be.

2.2.1 Benchmark Environment

Specifying the benchmark environment usually consists of listing the various components of hardware and software you will be using during the execution of the benchmark.

Depending on the purpose of the project, the conditions that have to be met could range from very specific (imposing the exact type, model, and configuration of the hardware components or the exact software level down to the latest bug fixes) to still rather vague (stating a price range only for all or part of the equipment to be used). In the latter case they need to become more precise as the project progresses.

In order to be able to later obtain the most meaningful interpretation of the benchmark results, however, all the details about the benchmark environment must be documented once they have been decided upon. There must not be any doubts about which equipment or software were used for a particular benchmark.

2.2.2 Benchmark Workload

In order to summarize the most relevant technical aspects of a benchmark, the term *workload* has been coined.

Workloads can be classified into different types, depending on the type of application, the amount and format of data to be stored and processed, as well as the method of processing. This also adds a further dimension to the classification of a benchmark itself.

Table 2 on page 10 contains the most common workload types used in benchmark projects and gives a short description for each of them.

Table 2. Workload Types

Workload Type	Description
Scientific and Engineering (S&E)	Usually involves manipulation of vast quantities of data by applying mathematical algorithms. It is often written in FORTRAN and uses parallel programming techniques. Classic examples would be weather prediction or modeling of dynamic systems.
Commercial Online Transaction Processing (OLTP)	Usually involves a database and many online users requiring fast response times to their queries in the order of seconds. This may well be a client/server application. These systems are often called mission critical or front office systems.
Commercial Batch	Usually involves a database and a long running task that needs sole use of the database to create new summary data, make large updates to the database or create reports. Often the batch runs at night on an OLTP system.
Decision Support (DSS)	Usually involves a database but only a few users that request large searches of the database to answer specific questions which are different every time and the response time is in the order of minutes or hours.
Internet Technology	Involves any of the new Web, Internet, Intranet, e-mail, e-commerce, Lotus Notes or object technologies and may involve a backend database.
Industry Standard	These are well known standardized computer benchmarks that are usually written and audited by an independent body. Examples are SPEC tests for CPU performance, TPC-C for OLTP and TCP-D for DSS.

Given the two tables above, you can now classify your particular benchmark by benchmark purpose and workload type. Some benchmarks include components of OLTP as well as Batch. The other workload types rarely overlap.

Examples:

- It is an OLTP benchmark to size a system.
- It is an OLTP and Batch benchmark against the competition.
- It is an Internet benchmark to prototype particular features of a certain product.
- It is a DSS benchmark to provide a proof of concept.

If this classification is used when talking to benchmark specialists, they will now have a clear understanding of the nature of your request and of the work involved. We will also use this classification in later chapters of this book to highlight features that are specific to different benchmark types.

2.3 Methods of Benchmarking

At the core of the benchmark project, there is the requirement to do some performance related measurements. In order to obtain meaningful results, it is not only necessary to specify the exact *conditions* under which the measurement has been taken place, but also to make sure that the proper *methods* have been employed.

In this section we will present the most commonly measured quantities and explain the methods that should be applied in order to measure them properly.

2.3.1 Runtimes

Benchmarks with workloads of the types Scientific & Engineering or Commercial Batch often require that runtimes for different jobs be measured. Although this does not seem to be particularly difficult at first sight, there are several things to consider:

- What exactly is the runtime of the job?

  In other words: What is considered to be the beginning of the job? When is the job considered to be finished?

  Sometimes jobs are submitted to some scheduling mechanism for later execution and the results are collected in a file and later sent into the submitter's electronic mailbox. Are we interested in the time between the submission of the job and the arrival of the output in the mail? Or are we only interested in how long it took from the beginning of the execution of the first statement of the job until the last statement was finished?

- Is there only one runtime?

  If we repeated the measurement several times, we would probably not obtain identical results. After several repetitions, we would have a *distribution* of runtimes. Which is the true runtime we are interested in? Should we take an average value, the best result, the worst? Sometimes there are a few measurements that produced values that seem to be much too high. Should we consider them any further or just throw them away?

There are no general answers to these questions. What might be the appropriate approach in one case might be totally inadequate in another. What is important, however, is that these points be addressed in the process of defining the benchmark objectives and that the participants in the project choose the method that suits their needs best.

2.3.2 Transactions

The central concept behind benchmarks from the Commercial OLTP class is the *transaction*, and it is also frequently utilized in benchmarks of other classes.

A transaction is a part of the application that is executed between a specific user input (usually clicking a mouse button or hitting the enter key) and the appearance of the application's response on the screen (usually displaying another window or some lines of text). New transactions can also be defined as a concatenation of previously defined transactions, which leads to having a *hierarchy* of transactions.

Real applications involve many different transactions. Typically, there are between about 40 and 2000 different ones. Often these are on different screens of the application and can be reached by a menu system.

Many application users will understand what you mean by different screens rather then transactions, but be careful because some transactions might involve more than one screen. Also note that these are often referred to as *business transactions* by benchmark people and this must not be confused with *database transactions* or SQL statements.

Typically, a business transaction (something like adding a new invoice or modifying an account) will involve one main screen (optionally other sub screens) with read-only database transactions containing multiple SQL statements to look up the details followed by the user committing the business transaction with further database transactions involving insert and update statements and a database commit:

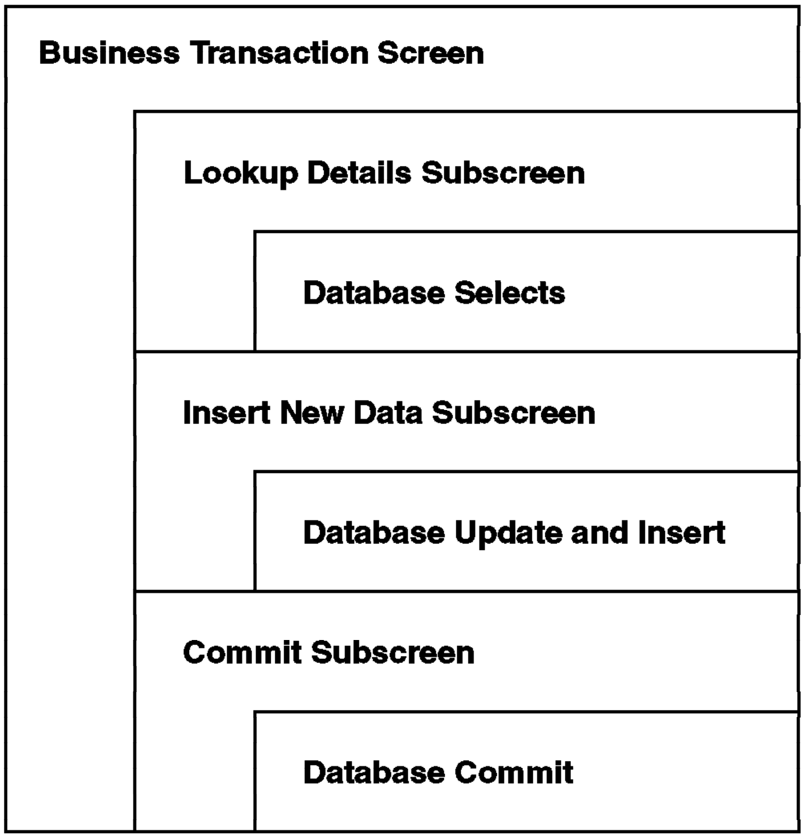

Business Transaction Screen
Lookup Details Subscreen
Database Selects
Insert New Data Subscreen
Database Update and Insert
Commit Subscreen
Database Commit

Figure 1. Transaction Hierarchy

Why are we highlighting these different terms? Because they can cause great confusion when people with different backgrounds are using the same term but are actually talking about different things. Do not assume everyone means the same thing by the word transaction. Precisely define which steps are included in each transaction you will use in your benchmark.

2.3.3 Response Times

In benchmarks using transactions, the quantity that is usually measured is the *response time* of the transaction. The response time is the time it takes to complete the transaction. This is the time between the two events defining the beginning and the end of the transaction. During this time the user waits for the application to respond to his or her input.

Response times are usually measured as a function of the number of users who are utilizing the application. Since all these users interact on the same system, response times are statistical quantities, so there is a distribution of response times for each transaction and every number of users. Generally, response times increase as you increase the number of users.

During a benchmark run, occasionally many users may hit the Commit key at the same time. The computer system will act on all these requests but they will take longer than normal to complete. For example, most of the time the system responds in less than two seconds but once an hour the response time is closer to five seconds. This is not worth failing the benchmark, but it should only be allowed to happen occasionally.

It is therefore appropriate to use *percentile* response times when stating a response time requirement for the test. The X percentile response time is the time that is larger than X percent, but smaller than 100-X percent of the response times measured. For example, if we use the 95 percentile, the result might be that 95 percent of the response times were less than 1.7 seconds and only 5 percent where longer than 1.7 seconds. There might even be a couple of response times in between the 4 to 5 seconds time, but these exceptions are normal. To get *every* response time below two seconds might require resources three times more powerful and would induce much higher cost.

2.3.4 Throughput

Complementing the response time information as a second quantity, *throughput* is measured in transaction oriented benchmarks. Throughput is the number of transactions that have been completed, divided by the time it took to complete them. This is also recorded as a function of the number of users utilizing the application. It can be used to calculate the time it takes to complete a given number of transactions by a fixed number of users. Generally, increasing the number of users increases throughput up to a maximum, then increasing the number of users further will decrease throughput.

This means that in a given environment for each transaction, there is an optimal number of users such that a given number of these transactions can be completed in the shortest possible time. Increasing the number of users beyond this point will not only increase response times, but will also decrease throughput and therefore increase the overall time needed to complete the transactions.

2.3.5 Remote Terminal Emulation

In any OLTP system there are online users at keyboards typing in information and reading the output at the screen. These terminals can be dumb green screens, X-Terminals or PCs. Benchmarks used to involve dozens of hired people, each with a terminal and a written script of what to type and how fast to work. The benchmark manager blew a whistle and everyone would start to work. On the second whistle, the users stopped and wrote down what they thought of the response time. This method of generating a workload for the benchmark had a couple of problems:

- The users got lost in their scripts and got stuck.
- The users worked too fast or too slow.
- It took many days to train the users to get it right.
- There is no accurate measure of response time, and users' opinions vary.
- It is costly.

Using this method requires people to do mundane and repetitive tests. Computers, however, are quite good at these types of tasks, and so as computers became faster and cheaper they replaced the people. This is how user emulations and workload generation tools were developed. In fact, computer generated workloads are far better than human ones because computers do as they are told, do the same thing every time, and can measure time more accurately.

To use such a tool, you need at least two systems: the *System Under Test* (SUT), which runs the application, and the *Remote Terminal Emulator* (RTE) which runs the emulation software. Both have to be connected, either through a network (such as Ethernet, token-ring or FDDI) or through multiple asynchronous connections (for example RS232, one line for each terminal to be simulated).

The steps involved in using their various components are as follows:

Capture user session: On the RTE, you capture a real user session by recording the keys pressed at the terminal and sent to the SUT as well as the output received from the SUT to be displayed on the screen.

Build a script: You build a script in a special format so the user session can be replayed from the RTE. This is done by sending the recorded key stokes to the SUT and then waiting for the reply that matches the expected output.

Note that many programs do not return exactly the same data every time. The fields might have different contents or include the time of the day. Therefore, the scripts usually have to be modified to account for this by selecting part of the reply which is specific enough to identify the end of the transaction, yet does not change each time the transaction is invoked.

Modify the script: In order to emulate more than one user, the scripts are further modified in a number of ways. If every emulated user typed exactly the same thing, then every user would be using the same part of the application and, even worse, the same part of the database. This would probably improve performance but is regarded as unrealistic and therefore as cheating. It can also cause major problems with database locking mechanisms. So the scripts are modified to:

- Make each user access different records by entering different strings into the application fields. This is usually done by allowing the tool to select the input value from a range or a valid set of values.
- Make the users add different records, again by selecting from a range or a valid set of different inputs.
- Make the users' typing rates and think times between consecutive transactions slightly random. This makes sure that the generated workload is slightly erratic and not at a perfectly flat rate (the computer generated workload can be too perfect, and a little randomness makes it behave more like real users do).
- Define the transactions in the script where user response times need to be measured.

Test the script: The RTE can now run the same script multiple times to emulate lots of users at the same time.

This needs to be tested to make sure the scripts work with larger numbers of users. Common problems include two emulated users accessing the same database record causing the second user to receive an unexpected record locked message. This confuses the script but this can be handled, or in most cases, avoided altogether by adapting the scripts accordingly.

Run the benchmark: Now you are ready for real benchmark runs emulating different numbers of users.

It is important to note that when a benchmark run starts, you must not attempt to get all the emulated users started at the same time. Logging in to a system and starting the application is a surprisingly CPU-hungry task. If, for example, 200 users do this at once then a large run queue is the result and some of the emulated users will not receive the expected responses in time, so that their scripts get confused.

It is, therefore, recommended that you stagger the user start times. For example, start 10 users every 5 seconds and slowly increase the number of users up to the number required. This is called *ramping up* the number of users in the first part of a benchmark test run. Fortunately, the user workload generating tools can do this easily.

Also note that benchmark response time results must only be taken after the ramp-up period has finished. This means the workload over the benchmark test run time will look approximately like this:

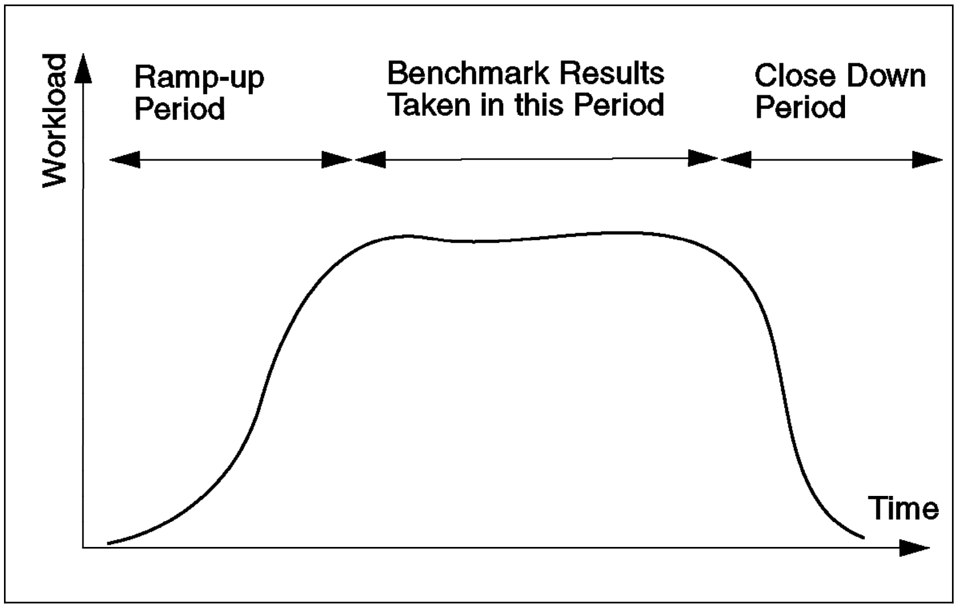


Figure 2. Workload Ramp-Up and Results Period

Make sure that during the measurements the RTE is not overloaded. Simulating too many users on a single RTE can produce invalid results. For example, if you increase the number of users on an already overloaded RTE, response time and throughput do not change because the RTE cannot further increase the frequency of attempted transactions. If this happens, you would need a more powerful RTE, or you could use multiple RTEs to distribute the load.

Generate report: Once you finish a benchmark run, the tool will provide a report of the actual response times of the transactions with some statistical analysis already applied. Typically you get the minimum, maximum, average and percentile response times for each transaction defined in the scripts.

2.4 Givens and Goals

As we have discussed in the previous sections, benchmarks should be designed in a way that clearly state the *purpose* of the benchmark, what the *conditions* of the benchmark are (Givens) and what *results* are expected from the measurements (Goals).

Depending on the purpose of the benchmark, there are usually only a few different sets of Givens and Goals that are typical and will be listed in the following paragraphs.

Competitive:

- Given is the price or a price range for the system to be tested, the workload, and the required response time or run time.
- Goal is to prove that the requirements can be met and exceeded. Since the results will be compared to the competitors' results that are usually unknown, the requirements have to be exceeded as far as possible.

Hurdle:

- Given is the price or a price range for the system to be tested, the workload, and the required response time or run time.
- Goal is to prove that the requirements can be met. Since there are clear success criteria, the benchmark project stops as soon as they are met.

Sizing:

- Given are a small number of different workloads and the required response times.
- Goal is to find a system configuration which meets the requirements. This usually involves testing different system configurations.

System Test:

- Given are a range of configurations, a set of transactions and the required response times.
- Goal is to find the maximum throughput for the different configurations and test the application. This usually requires a maximum system configuration.

Vendor test:

- Given are the configuration, workload and required response time to begin with. But any of these parameters are likely to be changed during the project.

- Goal is to prove the vendor’s ability to adapt to changing customer requirements quickly.

In order to reach these goals quickly and at the least possible cost, we have to think about how the tests should be designed.

2.5 Using Resources Effectively

Once the givens and goals have been identified, we need to think about the scope of the tests we should run during a benchmark project.

Assuming that the configuration to be tested is not yet completely defined so that we have some choices left (for example, how many CPUs or how much memory the system should have), we could think of testing different configurations.

At an early stage of the project, usually the transactions that should go into the workload to be tested are not yet defined. So one might think that testing as many of them as possible would be a good thing to do.

Did we already decide on how many users we should have in our tests? How often should they use the different transactions? Should they all have the same script to run or should there be different types of users?

So we might decide to have:

- *Three* different machines, having *six* different configurations of CPU and memory each
- *Four* different scenarios, in which different user types have different weights
- *Five* different workloads (in terms of numbers of users)

This would sum up to 3*6*4*5=360 different benchmark runs. If each of them would take one hour to complete, this would be 360 hours or 45 days (eight hours each). If it took two people at a daily cost of $1,000 each, the cost would be $90,000, which is probably not much less that the cost of one of the systems in the test.

This simple example shows that benchmarks can be quite expensive. But it also holds the key to improvement: we have to limit the scope of the tests to what is close to the expected outcome.

We could significantly reduce the cost of the project if we knew the outcome beforehand and could then prove that we were right by doing a single test. Although this is usually too much to expect, we should nevertheless try to come as close to this as possible by assessing all the information about the proposed environment we can get. This is why some people consider designing a successful benchmark to be more like an art than a science.

So we should find out, for instance, which are the most important transactions in the application. The 80-20 rule states that during 80 percent of the time only 20 percent of the transactions are used.

If we could narrow this down further by obtaining an estimate on the ″top ten″ transactions (or even less) and their share of the overall runtime of the application either by monitoring the application in a production environment or by making some intelligent guess, we could find a *single* scenario which contains these transactions according to their estimated weights.

We should also restrict ourselves with regard to the hardware configurations to only *one* machine with *four* different configurations. Finally, we could also drop one of the workloads and test only *four* different numbers of users.

This would result in 1*1*4*4=16 different benchmark runs which is much more reasonable than our previous number had been.

2.6 Organizing Results

During the benchmark runs, we will collect various result data. In order to easily identify which results had been obtained using which configuration (hardware as well as software) it is useful to utilize a scheme that prevents us from accidentally mixing up our data.

As a general rule, independent parameters should only be changed one at a time. We can, therefore, define different series of measurements which differ in one parameter only, which could be the number of users, for example:

- Series A: Machine 1, 4 CPUs, 1 GB Memory; 100, 150, 200, 250 users
- Series B: Machine 1, 4 CPUs, 2 GB Memory; 100, 150, 200, 250 users
- Series C: Machine 1, 8 CPUs, 2 GB Memory; 200, 250, 300, 350 users
- Series D: Machine 2, 8 CPUs, 2 GB Memory; 200, 250, 300, 350 users

This would be a total of 16 measurements. The files containing any results from these measurements should be placed in separate directories, one for each series, and the file name should contain the series as well as the value of the parameter that is varied. A file containing measurements of the CPU utilization during the test with 150 users from series B could be, for example, `CPU.seriesB.150users` and should be in directory `/results/seriesB`.

It is also important to put the definition of the series in an additional file in the directory `/results`, so we can easily look up which values the fixed parameters had in each of the series. Note that in practice there could be many more parameters to keep track of than in our example. For instance, those related to the database or the application. This is why we suggest you use the series definition instead of trying to code every parameter into the file name directly.

Since it is easy to make a typing error when copying results files, one should not rely on the name of the file alone, but always put an identification tag *inside* the file as well. This could either be the complete list of parameters or just the filename itself which contains all the information to exactly identify the environment in which the measurement was made.

2.7 Presenting Results

There are many ways to present results, and not all of them will lead to the same conclusion. Here is an example:

Let's say machine X took 10 minutes and machine Y took 15 minutes to complete a commercial batch benchmark.

Clearly, machine X is faster than machine Y. But how much faster is it?

Depending on how you do the calculation, you would get the following results:

- (Y-X)/X = (15-10)/10 = 1/2 = 50 percent
- (Y-X)/Y = (15-10)/15 = 1/3 = 33 percent

Which answer would you use? Maybe, if the competition's machine is faster than your own, it is only 33 percent faster, but if your machine is faster, it is clearly 50 percent faster?

We could also calculate the performance of the machines in terms of throughput (runs per hour). In this case machine X does six runs per hour and machine Y does four runs per hour.

Again, the results depend on how we calculate:

- (X-Y)/X = (6-4)/6 = 1/3 = 33 percent
- (X-Y)/Y = (6-4)/4 = 1/2 = 50 percent

This time, the results seem to be the other way around.

Since machine X cannot be both 33 percent faster than machine Y and 50 percent faster than machine Y at the same time, let us have a closer look at the facts and how we can express them in a less ambiguous way.

If we want to make a quantitative statement about how much faster machine X is than machine Y we should refer to a quantity that is positively correlated to the speed of the machines, that is, its value should be *higher* for the *faster* machine.

This is true for the throughput as we defined it above, but not for the runtimes. So we should compare the throughput of the machines by stating that

- the speed of machine X is 1.5 times (150 percent) the speed of machine Y

and

- the speed of machine Y is 2/3 (66 percent) the speed of machine X

Both statements are equally true, and there is no ambiguity so far. But doesn't the first statement sound like

- Machine X is 50 percent faster than machine Y

and the second statement like

- Machine Y is 33 percent slower than machine X?

The point to remember here is that whenever you start using relative numbers you should make clear what quantity is to be the common denominator. When we read statements like ″Machine X is 50 percent faster than machine Y″ we *implicitly assume* that the speed of machine Y is the common denominator because machine Y is used as a reference and therefore *interpret* it as being equivalent to ″the speed of machine X is 150 percent the speed of machine Y″.

We are still left with the possibility to talk about runtimes instead of throughput if we like the numbers more this way. But since runtimes are not positively correlated to the speed of the machines, we should not talk about fast and slow to avoid misinterpretation.

A proper way of stating the results would, therefore, be

- Runtime on machine X is 2/3 (66 percent) of that on machine Y
- Runtime on machine Y is 1.5 times (150 percent) that on machine X

Which could be translated into

- Running the job on machine X takes 33 percent less time than on machine Y
- Running the job on machine Y takes 50 percent more time than on machine X

which will be interpreted in the in the way we intended it to be by making the implicit assumptions about the common denominator, as explained above.

So if we want to stress the difference between the two machines (because ours is the faster one) we should use:

- Machine X is 50 percent faster than machine Y

or

- Running the job on machine Y takes 50 percent more time than on machine X,

whereas if we want to say that the difference is, in fact, not that big we could say:

- Machine Y is (only) 33 percent slower than machine X

or

- Running the job on machine X takes (only) 33 percent less time than on machine Y.

So we still have a choice between different statements from different points of view without unduly misleading the reader of our report.

Chapter 3. To Benchmark or Not to Benchmark

An old, wise and battle-worn benchmark person said the following:

- “Benchmarking is an art, not a science!”
- “Benchmarks take three things: experience, experience and experience.”
- “No benchmark at all is better than a failed benchmark.”
- “Doing a benchmark is like making love to a gorilla - you don’t stop when you get tired but only when the gorilla gets tired.”
- “Benchmarks are only as good as the planning permits.”

In this chapter we will explore:

- Some of the alternatives to benchmarks that you should consider.
- The size of a benchmark, so this is not under estimated.
- Risks associated with benchmarks.

This is so that you can choose whether to proceed with a benchmark or not. You also will know why the above bizarre sayings are so true for benchmarks.

Let’s learn more about what benchmarks are like by looking at a few case studies taken from real situations. This should highlight how you should be thinking and what is involved with a benchmark.

3.1 Case Study One - Benchmarks Are Bigger Than People Think

Your manager tells you to:

- Build and configure a new large production system from limited information.
- Design and implement the database for maximum performance and load the database up to 90 percent full with data that is currently in the wrong format.
- Install the largely untested application.
- All the users would arrive on day one and there is twice as much to do as they originally expected.
- You have to prove it has sub-second response time by measurement.

Your response might be unprintable or you might say “that looks like five to six months work for a team of four people.″ But your manager retorts, “Oh come on, it can’t be that hard, I promised you will do it in two weeks starting on Monday. You have to finish or your job is at serious risk!”

Clearly, doing all this in two weeks is impossible - but this is exactly what is expected in a benchmark.

Lessons from Production Systems

- To benchmark a system you have to build a complete system.
- You only get time to build it once which means planning every detail to avoid time consuming mistakes.
- Benchmarks often start with poor applications and data.

3.2 Case Study Two - Benchmark Precision Can Be Very Tricky

Let us take a very simple and straight forward benchmark that cannot possibly go wrong. Our example:

A 20 MB file transfer is used as a benchmark. The performance of machine A and B will be compared. Machine C is used as the destination. Machine A completed the file transfer in 10 seconds. Machine B completed the file transfer in 45 seconds.

Question: Do you have enough information to compare the performance of the two machines A and B?
Answer: Yes, the results are very clear.

Question: Which machine would you buy?
Answer: Machine A, of course.

But you have made all sorts of assumptions, so here are some additional facts:

- Machine A averaged 98 percent CPU busy during the transfer.
- Machine B averaged 28 percent CPU busy during the transfer.

Conclusions: Machine B might be able do three transfers at once but machine A could not. In production, is a single transfer or multiple transfers more likely?

Some additional facts:

- Machine A was configured to use a 1500 byte packet size and binary data.
- Machine B was configured to use a 500 byte packet size and text data.

Conclusions: The test was rather different on each machine, we don't know if the results are valid.

Additional facts:

- Machine A had the 20 MB file cached in memory from a previous run.
- Machine B had to access the file from disk.

Conclusions: The benchmark environment must be exactly the same for each test (even on only one machine), even hidden things like a cache, otherwise results cannot be compared. Some people might consider using the cache as "cheating" but it is normal operation for database systems. A ruling must determine whether a particular technique should be used or not. If advanced techniques like this are to be used, some way of making sure the same starting point is used for each test run needs to be established.

However, sometimes it is appropriate that the benchmark test is optimized to run most effectively on different machines. Thus making "best use" of the different features of each machine. In the case above, if one machine can use a cache but the other cannot, then it is fair to have the test run differently on the two machines.

Conclusions: This was a very simple benchmark, but we still have no idea which is the better machine since the performance issues have become very confused. It is only experience that can avoid these sorts of mistakes in real benchmarks.

Understand Everything Before You Start

- Know and define the production system operation details.
- The benchmark person must decide the configuration to be used across all tests and the results must include the precise configurations used.
- You need to know every performance trick in the book.

3.3 Case Study Three - The Price Dimension

Let's suppose:

- Machine N does 40 transactions per minute and costs $100,000
- Machine M does 30 transactions per minute and costs $50,000

Which machine is the best?

Answer is *both*:

- Machine N can do more transactions per second.
- Machine M does more transactions per dollar.

Note: This happens quite often in competitive benchmarks when the benchmark goals are unclear and the vendors bid different size machines.

Which vendor wins the benchmark?
Answer: That depends on the customer:

- If the customer wants the highest performance level, then machine N wins.
- If the customer wants good price performance, then machine M wins.

In fact, in this case, knowing your customer is more important than a good benchmark result.

Know What Your Customer Wants

- It must be clear in the benchmark objectives what the customers wants from the benchmark. Is it price or throughput?

3.4 Case Study Four - One Primary Objective

Let's say the benchmark results are as follows (for simplicity there is only one response time and only two configurations):

Table 3. Mixed Benchmark Results

Test	Machine F	Machine G
OLTP test S	10 TPM	12 TPM
OLTP test T	1000 TPM	800 TPM
Batch run	3 hours	2 hours
Load test	30 minutes	60 minutes
Price	$100K	$100K
Note: TPM is transactions per minute		

How do we decide which is the better machine?

- The customer asked for all of this information and said it was all critical for making a decision about which machine they would buy.
- One machine is better at some things but the other is better at others.
- There is no way to compare a batch run of two hours with an OLTP result of 1000 TPM.
- Will the measurements be added, averaged, weighted or normalized?

- What if a particular vendor only completes part of the benchmark, but with very impressive results?

The answer is, again, it depends on the customer and how they rate the importance of each of this test numbers. In fact, most customers would actually like two competitors to present results like this because:

- It shows that both machines have roughly the same price/performance.
- Which ever the customer chooses they will not get a bad machine.
- They can decide which aspect is most important and feel comfortable.
- They can actually chose which vendor they prefer (for non performance reasons) and then find a justification based on these results.

Lessons: Try very hard during the benchmark to make a good impression regardless of the stress and pressure of the benchmark because it could make the difference. In these cases the vendors will argue that they in fact won the benchmark but they may be better to call it a draw and market better systems management, industrial strength or a better partnership.

Your Prime Objective

- The primary objective should decide the benchmark success.
- Keep secondary objectives as “nice to have”.
- This focuses the team on the success issues

Benchmarks Are an Art

By now you should have realized that with benchmarking we cannot test every possible combination. We must compromise in many areas and use many assumptions. This is why benchmarking is an art and not a science.

3.5 Case Study Five - Why Benchmarking is Difficult

In essence, there are only seven things you need to do a benchmark:

1. Goals
2. Hardware
3. Software
4. Data
5. Workload
6. Skills
7. Time

The basic benchmark problem is that it is very hard to find information and resources. We have already seen in the case studies that a benchmark needs to create a production system and a fixed repeatable workload in a very short period of time. This is hard enough. But the chances of something going wrong are large, and one mistake will often hold up the entire benchmark. If you examine the list, it becomes clear that the first five must be performed in that order. The only way to avoid mistakes is careful and thorough project planning to remove all nebulous statements, misunderstanding or gross under estimations of all these aspects and to make sure all the right parts come together at the right time. This is what the majority of this book is about. We have to be precise about every aspect of the benchmark:

Goals	Have to be clear and numeric with priorities.
Hardware	Has to be there, working, configured and very often more is needed at short notice.
Software	Has to be exactly the right versions, with fixes, configured correctly and on the right media.
Data	Has to be realistic, complete, consistent, needs to be created/loaded quickly, in the right format and on the right media.
Workload	The application must work and the workload has to be defined precisely and be reproducible including what screens and at what rate.
Skills	A large number of different skills are required tat the expert level and all must be available.
Time	There must be realistic time frames with contingencies.

Murphy's law applies especially well to benchmarks. Murphy's Law states "if it possibly can go wrong, it will" (also know as "butter side down"). At least, one (if not all) of the above parts of a benchmark will go wrong. This can be guaranteed. These unexpected problems will hold up everything. Benchmark plans never have enough contingencies because we have already cut down the number of things that would be nice to try in a benchmark.

How are these things overcome?

- Hard work and always thinking ahead to the next phase.
- Long hours and even weekend working.
- This results in "benchmark burn out" - because of the high pressure and long hours. People should not be expected to run one benchmark after the next without recreation time in between them.

- Calling in favors from expert friends and those with hardware resources.
- Experience pays great dividends because experience does not have to guess at the solution to a problem.

.

What are Benchmarks Really Like?

- Problems and long hours are guaranteed.
- Technical experience really counts when the going gets tough.
- Emergency benchmarks resources can be very hard to find.

3.6 Case Study Six - The Good, the Bad and the Ugly

Below are a few examples of good, bad and awful benchmark definitions. Note that most of these are taken from real life cases. The point is that only the precisely defined benchmark is likely to be successful.

Table 4. Good, Bad and Ugly Benchmark Definitions

	Good	Bad	Ugly
Goals	200 users @ 10 transaction per second.	It is 341 users but we don't know how fast users will work.	Just prove it will work when it is in production and give me a guarantee.
Hardware	J50, 4 GB RAM, 17 GB SSA Disk, FDDI	F50, or R50 or may be...	You tell me!
Software	AIX 4.3, PTX, C compiler, Oracle 7.3.3.3 on CD-ROM.	I think it is Oracle but I'm not that sure.	This has not been decided yet; could we try three different databases?
Workload	Tested Application, Screens 1,2,3,4,5 used equally between 200 users.	We hope to have the application ported before the end of the benchmark.	The application has 1004 screens lets test all of them.
Data	Exists and is tested, we will provide it by DB Export and supply a sample tape.	We have it but it is on a 30 year old machine and it is a flat file in EBCDIC.	Sorry it got corrupted. Can't you generate some for us or I could write a shell script to make some?
Skills	DBA and Application specialist full time, both know AIX.	Sorry the DBA and APP specialist can only be used for 4 days and there is no telephone help.	I was told you had the skills so why should I help?
Time	Two weeks with a further one week contingency.	FAX the results on Friday at 4pm for a decision by 5pm or you get the blame.	Can't you do three weeks work over the weekend? I thought you were committed to this!

The more your benchmark details are like the Bad or Ugly cases the more problems you are going to encounter as it proceeds. This introduces the concept that benchmarks can be risky. This risk of a benchmark failure is constantly on the mind of experienced benchmark people. The benchmark process is there to eliminate these risks and even stop a benchmark that appears likely to fail.

Ask yourself: What is the worst that can happen?

- The performance was below the expected results.

- The hardware, software, database or user emulation tools failed or could not be made to work at all.
- Due to lack of appropriate skills, wasted time or lack of planning the benchmark ran out of time and failed to document any results.

Whatever the reason, the effect on the customer can be:

- Total loss of confidence in the vendor.
- An opportunity for the competition to attempt a benchmark.
- Major impact in the customer account for a long period of time.

In other words, a complete disaster. If it is clear that the benchmark has no chance for success, you should convince the customer that it is in their best interest (that is, money and people time saved) to not attempt it.

In B.9, "Risk Assessment Form" on page 139, there is a risk assessment form that can be used to evaluate how risky or safe a benchmark is likely to be. If you think you have a problem benchmark this should help you by documenting the risks.

Cancel Rather Than Fail

- A failed benchmark can do a lot of damage but a successful benchmark is only part of the selling process.
- Benchmarks are complicated and can only be as good as the planning permits.

3.7 Alternatives to Benchmarks

Using the classifications Benchmark Purpose and Workload Type that were established in Chapter 2, "Benchmark Basics" on page 7, we can look at some of the alternatives to benchmarks. The point of exploring alternatives is that they all have less risks and are less costly than benchmarks. Some alternatives to benchmarks also have the benefits that the customer is more involved, works in partnership, and starts building their production system. This means that they have already committed to the purchase of a solution at an earlier stage. Alternatives are all quicker to implement than a benchmark, and are used for the following reasons:

- Quicker to get result
- Less expensive in resources
- Less demanding of equipment
- Less risk of failure
- Building up a permanent partnership between vendor and customer

Below is a summary of the alternatives:

Estimates	Many times the customer needs to find out the approximate size of a proposed system to get budget approval before serious purchasing is started. This may be achieved by using previous application knowledge and the typical performance from existing, similar systems. There are sizing tools available that can assist in using what knowledge is available to determine suitable estimates. A good use of a spreadsheet can be useful. In addition, estimating is the sizing method proposed and advised when the application is well known. Most application vendors with large numbers of installed systems have gained sufficient understanding about the product and the way it behaves with a wide variety of workloads so that they have a sizing model to calculate a configuration.
Capacity Planning	Third-party capacity planning tools, Best/1 in particular, (see Appendix C, “Tools Used in Benchmarking” on page 151) can be recommended from experience and can be used to capture data on a currently-running, similar system. Similar means a system with the same application, workload and similar size. This data is then analyzed and categorized into the model of the system built into the tool. Then you can change the workload or system characteristics and request the tool to predict the new systems performance. This can be used to avoid benchmarks or to recommend upgrades to existing systems. However, the greater the difference between the captured data and the prediction, the greater the errors introduced.
Shared Reward	This is a joint agreement between the customer and vendor where they work together to determine the best guess or estimate. There are unknown factors because the application or environment is new. Therefore, the customer puts in the basic system with the understanding that if the estimates are low, the vendor will quickly upgrade the system as demand increases on the system.
Site Visit	If the main purpose of the benchmark is to satisfy the customers confidence level in the proposed configuration, then getting the customer to visit a similar production system can address this issue in a

	better manner. The customer also will learn much more about the problems and benefit from the solutions already used. They also know that they are not using cutting edge solutions.
Reference Site	If the benchmark is to prove that the proposed configuration is roughly adequate, then a reference production system site can address this issue. This is typically is used with well-known standard and popular applications, where each 200 user system will probably be very similar in configuration.
Prototype	In this case, there are usually new or unknown factors involved with the proposed solution. This makes a benchmark extremely hard because the environment and configuration are difficult to determine. Software components might not work together or might not be efficient at the time of the benchmark and the application may still be at the prototype stage. If the customer has access to a reasonably-sized machine for further testing, then these issues can be explored. Occasionally, the machines chosen to be used at the benchmark center are not be appropriate, since prototypes can take longer than normal benchmarks. Therefore, loaned or hired equipment may be desirable.
Testbed Service	This denotes a situation where the benchmark center supplies the equipment only and does not undertake any responsibility for the benchmark, other than seeing that the equipment works. The customer of the testbed service takes full responsibility for defining the benchmark, obtaining the skills and running the benchmark. This is typically used by business partners and independent software vendors.

You need to be sure that none of the above is suitable for your circumstances before starting a benchmark.

Table 5. Benchmark Alternative by Purpose

Benchmark Purpose	Alternatives	Benefits
Competitive	Benchmark may be unavoidable.	N/A
Hurdle	Reference site and site visit.	Quick, cheap and lower risk.
Sizing	Estimates based on previous application knowledge and sizing tools.	Quick, cheap but with some risk.
	Capacity planning tool using captured data on currently running similar system and then used for predicting the new system's performance.	Quick, cheap and lower risk.
	Business partner and ISV could use a testbed service.**	Quicker to schedule as no skills from the benchmark center, Risks taken by the B.P.
	Shared reward and return deals where the best guess is installed and upgraded as necessary.	Quick and lower risk.
System or Soak Test	Testbed service**	Quick as no skills from the benchmark center, lower risk for benchmark center.
Proof of Concept	Prototype at customer site on loan or hired equipment.	Quick and lower risk.
Vendor Test	Prototype at customer site on loan or hired equipment.	Quick and lower risk.
*** **Testbed service**: see explanation in text		

3.8 Qualities of a Good Benchmark

If you are writing a benchmark test then you should consider the following list before you start thinking about the hardware and software requirements. This should make your benchmark more useful and simpler to get the results you need.

Relevance	Does it represent important features of the real workload?
Specificity	Does it primarily test what it is intended to test?
Measurability	Is it easy to quantify the results in a meaningful way?
Repeatability	Is it easy to replicate the results of the benchmark by rerunning it?
Scalability	Can the benchmark be run on configurations of different sizes and still provide meaningful measurements?
Portability	Can the benchmark be easily run on different platforms or on different vendors machines?

Chapter 4. Benchmark Process Overview

This chapter is an overview of the benchmark process we describe in this book and the flow of information in it. The diagram below summarizes the process:

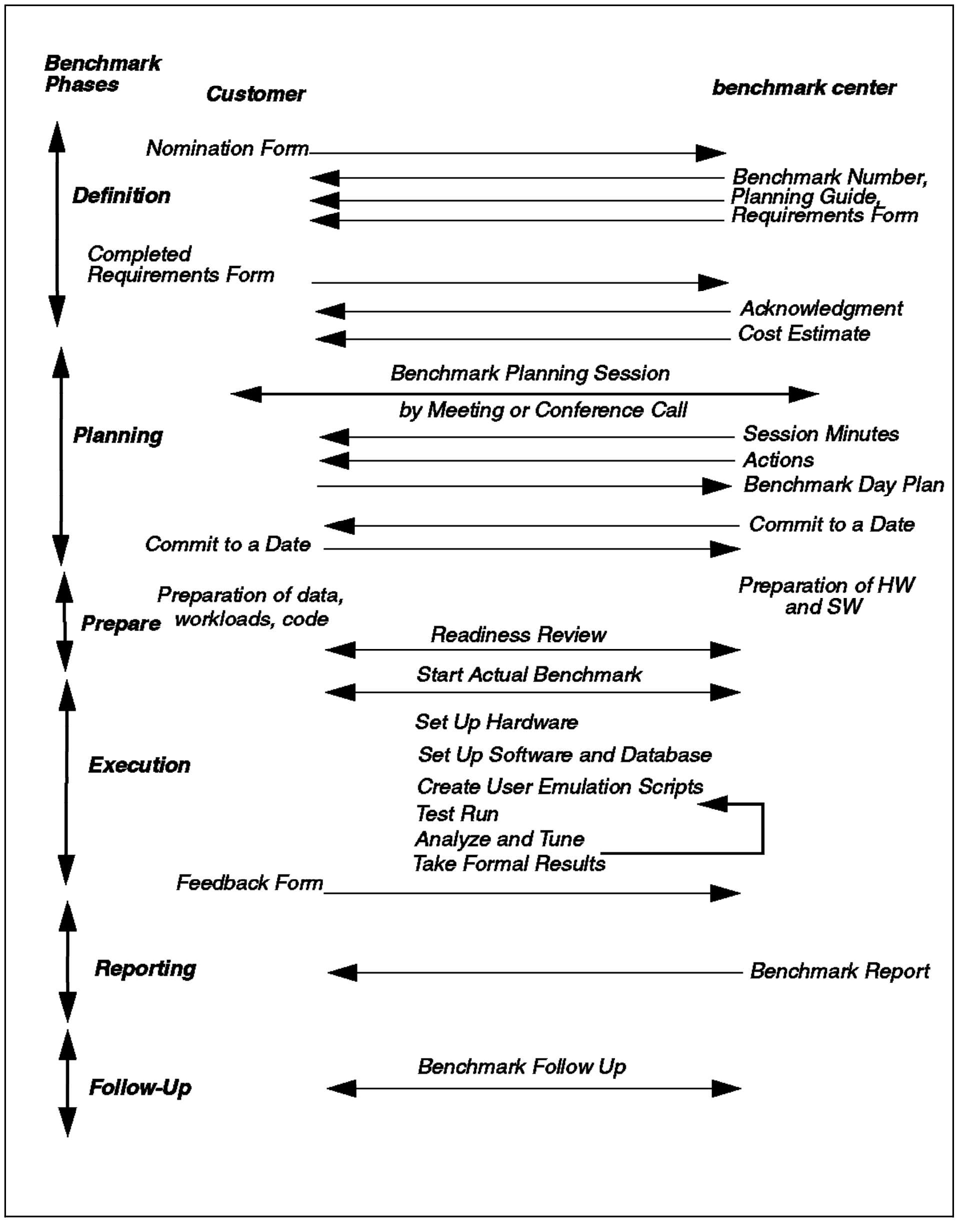


Figure 3. Benchmark Process Overview

The left edge of the diagram are the names of the six phases of a benchmark which are described in much greater detail in the rest of this redbook. Each of these phases is vital for the success of your benchmark. The following section goes through each of the phases of the benchmark and describes what takes place in each phase at a high level.

In this explanation of the benchmark process the term customer is generic for any person requesting a benchmark. This could be a real customer, a business partner or the marketing or sales account team requiring a benchmark as part of a bid or tender.

4.1 The Phases of the Benchmark Process

Each of the phases of the process are summarized below.

4.1.1 Definition

Although the customer and benchmark center might already be in contact with each other and talking about benchmarks in general, the first formal part of the process is the completion of a benchmark nomination form. A sample of such a form is included in B.1, "Sample Benchmark Nomination Form" on page 113. This form contains the most basic information about the benchmark and should not be hard to complete. It simply needs:

- Contact names, addresses, e-mail address and telephone numbers
- The business case, used to prioritize benchmarks
- Benchmark objectives and success criteria
- Any available technical information

Once this form is forwarded to the benchmark center, it should be quickly acknowledged and they should send you:

- A benchmark project number that is used for tracking the benchmark request.
- A benchmark planning guide to explain the process, assist you to developing the benchmark plan and ensure that all technical issues are addressed.
- A benchmark requirements form which is a document in which all the technical aspects of the benchmark must be clearly stated.

First, read the benchmark planning guide so that you understand the process and how benchmarks are organized. Then start work on the benchmark requirements form, which covers many technical aspects of the benchmark. Have a technical specialist complete the answers or work them out in discussions with the customer. Answers to all of these questions are required before it can be returned to the benchmark center. If you are having problems with particular questions you may want to talk to the benchmark center; they would rather talk you though questions than have you waste time or answer incorrectly.

See Chapter 5, "The Definition Phase" on page 47 for the full details.

4.1.2 Planning Checklist

Once completed, the benchmark requirements checklist is returned to the benchmark center, and this should be acknowledged. The benchmark center now has the information required to access the benchmark request and can schedule a benchmark planning meeting. This meeting should be attended by:

- The benchmark sponsor (the person paying) to state the prime objectives and to ensure the benchmark answers the fundamental question.
- The senior technical specialist involved in the benchmark.
- The database administrator, if a RDBMS is involved.
- The application specialist who will install and configure the application.
- The data owner who understands the database and how it is to be created.
- The benchmark representative from the benchmark center.
- Anyone else that might be useful.

The agenda of this meeting is discussed in B.2, "Sample Benchmark Planning Meeting Agenda" on page 115. The meeting should resolve all the outstanding questions, technical or otherwise.

As a result of the planning meeting, the session leader will send the following items:

- Minutes of the meeting and all agreements made.
- A list of the outstanding actions that must be addressed before the benchmark starts.

The customer is expected to prepare the day plan (what happens on each day of the benchmark) and send it to the benchmark center.

If the planning meeting fails to resolve all the requirements because there is missing information, then another meeting will take place. Once completed, the benchmark center will schedule the benchmark within its hardware and skills constraints and offer a benchmark date to the customer. If the customer can allocate the resources, this should be confirmed in writing.

See Chapter 6, "The Planning Phase" on page 55 for more details.

4.1.3 Preparation

Next, both the customer and the benchmark center must begin preparing for the benchmark. The benchmark center will prepare the hardware, some software and the network. The customer will prepare the application code and data, the database, refine the workload and any other software. During this preparation phase there may be a readiness review to make sure both parties can complete their preparations in the time available. The customer may also make travel arrangements for hardware, software, data and people to get to the benchmark center.

See Chapter 7, “The Preparation Phase” on page 65 for more details.

4.1.4 Execution

In this phase, the benchmark begins and runs to completion. This is the phase of a benchmark that is clearly the central part of the project. During this execution phase the bulk of the technical effort is used to create the benchmark system, establish the data, generate the workload and run benchmark tests. The point of the three previous phases are to minimize the problems and maximize the usefulness of this phase. Also, the system and database performance is analyzed and tuned for performance in the execution phase to make sure the best results are obtained. Very careful time management needs to be used to ensure that steady progress is made and the problems that do arise are addressed effectively with little wasted time and effort. At some point the decision has to be made that the benchmark has achieved success and can be completed. This can be caused by running out of benchmark time, exhausting all tuning options or reaching the success criteria.

Near the end of the phase the final benchmark tests are run and the formal results are taken. These are used in the reporting phase.

Most benchmark centers like a feedback form from the customer to learn from the experience and make improvements in the benchmark center and in the benchmark process.

See Chapter 8, “The Execution Phase” on page 73 for more details.

4.1.5 Reporting

Shortly after the benchmark, the vendor analyzes the results data and the benchmark report is written. Benchmark reports usually include a management summary, full details of the hardware and software setup, system customization, database data, the application, workload, performance results, graphical performance charts of machine utilization, a list of objectives and a description of how they were met. This benchmark report is then presented to the customer.

See Chapter 9, “The Reporting Phase” on page 93 for more details.

4.1.6 Follow-Up

After the report is presented, there are a number of activities that are performed to maximize the benefits of all the hard work involved in the benchmark. These include benchmark feedback to make sure the benchmark center and the process are improved, marketing and technical tips that might help other situations, filing the benchmark details and all documentation and returning hardware and software items to their owners.

See Chapter 10, “The Follow-Up Phase” on page 103 for the full details.

4.1.7 The Importance of Each Phase

As a guide, the relative importance of each phase is highlighted below by amount of benchmark management effort (man-days) typically used:

Definition	15 percent benchmark management effort
Planning	40 percent benchmark management effort
Preparation	20 percent benchmark management effort
Execution	20 percent benchmark management effort
Reporting	5 percent benchmark management effort

This highlights the importance of correctly defining, planning and preparing for a benchmark before executing it. For example, if the definition, planning and preparation are not finished and documented, the benchmark has a 75 percent chance of failure!

The following list details the percent mix for technical effort (man-days) typically used:

Definition	5 percent technical effort
Planning	20 percent technical effort
Preparation	10 percent technical effort
Execution	60 percent technical effort
Reporting	5 percent technical effort

This highlights that there is significant technical work in the planning phase and the execution phase.

4.2 The Process to Use

There are many benchmark centers around the world and each will follow their own version of the process and have their own forms.

In the course of writing this benchmarking redbook, the processes of many benchmark centers in the USA and EMEA were investigated and a surprising number of similarities were revealed. About 90 percent of the processes were common and the differences were insignificant.

Important! Use the Correct Process

You should check the process in your country and local benchmark center. Remember, there will be minor variations.

In particular, there will be small differences in terms and forms.

The forms in Appendix B, "Checklists and Sample Forms" on page 113 will give you a good idea of the types and technical levels of questions that you will be required to answer during the process.

Chapter 5. The Definition Phase

The definition phase of the benchmark project may be the most critical since decisions made in this phase will have a direct effect on the entire scope and complexity of the effort.

This is the time to decide exactly what the benchmark will consist of and what workload will be used for testing. This will then dictate factors such as equipment, software, skill requirements, areas of responsibility, the *time line* for preparation activities, and ultimately the start and end dates of the project. All of this of course has a direct bearing on the cost of the benchmark.

In this chapter we will discuss:

- Defining objectives and success criteria
- Defining the environment to be used for testing in terms of required:
  - Hardware
  - Software
  - Skills
- Defining the workload to be used in the benchmark in terms of:
  - Amount and format of data
  - Type of processing applied to it
- Defining measurement of benchmark results and the tools to be used.

The topics covered in this and subsequent chapters will use terms and procedures referenced in Chapter 2, "Benchmark Basics" on page 7. Therefore, the discussions in this chapter assume prior knowledge of the terms used.

5.1 Defining Benchmark Project Objectives

The first activity that should be undertaken is to have the customer set forth one or more statements that clearly and concisely tell what that want to accomplish by doing the benchmark. Even though this sounds like an inconsequential activity at first glance, in practice it is often quite difficult to do. Creating a good objective statement does the following:

1. It forces the participants to verbalize their thoughts, which often leads to intense discussions of just what is important versus what is not.
2. It helps orient (level set) all participants, so that from this point forward there is a common understanding of what should be accomplished.
3. It allows for prioritizing multiple objectives should there be any, so that the most important activities can be emphasized.

4. It sets the tone for the size and scope of the project, both of which have a direct bearing on the complexity, which is related to the cost.

After the objective statements have been created, the originator should set up a review of them with someone who is familiar with the proposed project, to get an independent validation of them. The reviewer may be the person who will act as the technical focal point for the project, the vendor marketing representative, the systems engineer (SE), or some other peer reviewer. The following items can be used as a starting point for the validation process.

- Do the objectives *accurately* and *adequately* describe what results are expected from the project?
- Are they *reasonable* when matched with the intended use of the new system?
- Are the objectives *attainable* considering the resources of time, skills, equipment, and so forth that will be available for the benchmark?

5.1.1 Sample Objective Statements

The following list contains some examples of both proper and improper objective statements.

Refer to 3.6, “Case Study Six - The Good, the Bad and the Ugly” on page 31 for more details.

Proper Objective Statement Examples:

- Complete all required *Batch* jobstreams for the *xxx* database application within a six hour wall clock timing window.
- Support 500 concurrent end users performing a mix of *Online System* transactions with a 3 second or less mean average response time at or below 95 percent CPU utilization. 95 percent of the response times must be less than 3 seconds.
- Support 500 concurrent end users performing a mix of *Online System* transactions with a 6 second or less mean average response time at or below 95 percent CPU utilization while concurrently running the *Batch Backup* jobstreams.
- Support 500 concurrent end users performing a mix of *Online System* transactions with a 6 second or less mean average response time at or below 95 percent CPU utilization while concurrently running the *Batch Backup* jobstreams on a hardware configuration costing less than $100,000.

Improper Objective Statement Examples:

- Support 500 concurrent end users.
  - This statement does not state performance objectives for supporting the 500 users and, therefore, is nebulous.
- Perform equal to or better than a XXX Model YYY machine. (XXX means an OEM competitor's machine.)
  - This statement does not quantify the performance parameters of XXX.

5.2 Establishing Success Criteria

Defining the success criteria for the project is as important as defining the objectives, and it is often undertaken as an integral part of the objective definition.

The criteria for the successful completion of the benchmark is an item that is frequently assumed to exist when in fact it does not. To prevent erroneous perceptions from being made and in the interest of good communications as well as good benchmark planning, a clear statement of what the customer views as success of the project should be made. This will convey proper expectations, and provide guidance to the technical participants as to how much tuning is required and when adequate tuning has been made.

However, sometimes there are certain pre-existing success bars, such as in the case of a government bid which has success criteria spelled out that cannot be changed. In this case you might have to develop the objective statements using these specifications as a starting point.

There are other somewhat less important objectives that the customer would like to achieve if time and resources permit, however, they are not critical to the success of the benchmark. In this case there may be a single objective statement that defines which objective must be met for the project to be called a success, followed by subsequent objectives which list (by priority) the additional objectives that it would be nice to achieve if resources permit. For example, the following set of statements might be used:

1. The benchmark will be considered successful when 500 users are logged in and executing accounting tasks with mean average response times of 3 seconds or less with CPU utilization of 90 percent or less on an RS/6000 model S70 with 6 processors and 8 GB of RAM.

2. If time permits, the above equipment configuration will be run with 600 and 700 users respectively and end user response times, whatever they may be, will be recorded.
3. If time permits, lesser amounts of RAM and fewer processors will be tested with 500 users to determine if there is a more economical processor/RAM configuration that will also yield 3 second or less response times at 90 percent CPU utilization.

In the example you will notice that the first objective statement lists the success criteria, then the two additional objectives are listed as things that it would be nice to do if time permits. It is assumed that the second item is less important than the first but more important than the third.

5.3 Environment Definition

The benchmark environment will naturally consist of the required hardware and software, but the environment actually goes beyond that and includes items that need to be considered at the definition phase and completely defined at the planning phase. What items might be included? While not all-inclusive, the following things should be considered:

- What hardware equipment will be required for this project?
- What software will each piece of hardware require?
- What skills will we need to have to insure our success?
- How will the executable code be serviced? Will it reside on the test system or will it be provided by a code server?
- Where will the data reside?
- What parameters will need to be set up before each test run?
- When will our timing window be opened after test execution start?
- What clean up activities will we need to do before the start of each test?

All items that are required to insure that each test starts with the correct testing environment should be included here. The planning phase includes work on a detailed document that describes how these items are used.

5.3.1 Defining the Workload

Defining the workload to be used determines at a high level the flow of testing steps to be employed. A correctly-defined workload should have the following characteristics:

It should encompass the *intended real use* of the system (applications, terminals, tasks, and so forth).

It should consist of a *meaningful subset* of the *real workload.* What this means is that it is seldom practical to try to use all tasks in all jobs in a production environment as the benchmark jobstream. This then presents a challenge for the planners. They must decide which tasks/jobs are important enough to be included in the test workload and which ones can be excluded. The jobs that are included should be:

- *Organized* in a logical manner. That is, if one job loads data that is later queried or manipulated, then this loading job must logically be executed before dependent jobs. Likewise if some jobs change the data such as by doing updates, deletes, and so forth, then care should be taken that a subsequent task will not be searching for the data that was previously deleted.
- *Representative* of the total production environment. What this means is that when all the work is completed for defining the benchmark workload, it should be reviewed and the following points should be considered:
  - It should be a workload that can be *implemented* for benchmark testing purposes without requiring an unmanageable work effort.
  - It should *support the objective(s)* statements we defined earlier. If it doesn't support those objectives it is not the proper workload.
  - We should be confident that if we use this as our test workload and we are able to achieve the results stated in our acceptance criteria, the actual production system we install based on this test will perform as we predict. If *not,* then this workload is not adequately representative of the real world and more work is needed.

We should decide whether the tests will be purely *batch* in nature, or if they will be designed to measure end user response time (interactive testing), or a combination of the two. This may also be an OLTP client/server test, and if so it may employ only clients querying a data server (two-tier), or it may have end users being handled by a terminal server, which in turn will query the data server (three-tier).

Whatever the workload definitions it should be thought of as having two parts, the data, which is static in nature, and the processing (batch or interactive, for example) which is dynamic in nature.

5.3.2 Batch Workloads

If it is determined that a batch workload is to be tested, either alone or in conjunction with other workloads, some of the following issues may be important:

- What is the beginning environment that must be set up before each test run?

- What will or will not be included in the testing, such as file open overhead, code service methodology, and report writing?
- How many jobs will be run?
- Will we run a single copy of each job sequentially or run multiple copies across multiple CPUs? Can we parallelize any of our jobstreams to take advantage of the SMP architecture?
- Do I expect that running 6 copies of a single job on an SMP machine with 6 processors will complete in the same amount of time as a single copy run on a single CPU? Why might the times vary?

5.3.3 Interactive Workloads

Some considerations for interactive workloads are:

- Insuring a correct starting environment each time such as in running batch work with the exception that the tasks required to restore the environment get more numerous and complex for these types of benchmarks.
- The number of end users that will be emulated on each RTE machine should be planned so that an overloaded terminal emulator machine does not inject errors in your measurements.
- You should decide whether to use a two-tiered or a three-tiered environment if doing OLTP testing.
- The tasks necessary to collect testing output on disk should also be considered.

5.3.4 Combination Workloads

The customer may find it advantageous to do a combination of both the batch as well as the interactive testing for the following reasons:

- There might be a requirement to see what the impact to *real-time* response time would be in the event that overnight batch processing must be run concurrently with their online application. (In other words, assuming that equipment problems occurred during the allocated batch window).

- There could be other reasons for running batch and real-time concurrently, which would require testing them together.

If you plan on testing combined batch and interactive workloads, you should use both preceding sections as a starting point and also consider any other items that might be unique to your benchmark.

5.4 Defining Measurement Metrics

To minimize the chance of collecting either incorrect or inadequate performance data for your project, you should consider the following:

1. What will be measured? (What metrics will be used?) The answer may be one or more of the following.
    - Elapsed wall clock time (Often used for batch tests)
    - End user response times
    - Transaction rates (typically data base oriented)
    - CPU and/or I/O loads
2. What system oriented data should be collected?
    - CPU utilization (through `vmstat` or `iostat`)
    - Disk I/O activity (through `iostat`)
    - Paging activity (through `vmstat`)
3. When and how will data be collected?
4. Who will collect and archive data and be responsible for generating performance reports?
5. Where will results data be stored?

5.4.1 Defining Measurement Tools

It does no good to decide on success criteria without some discussion of what tools will be used to collect results. This could range from something as simple as a stop watch to utilization of multiple sophisticated tools. The important thing in the definition phase is to agree not only on what the expected results will be but also what tools will be used to provide results data.

5.4.2 Defining Results Data Requirements

As you begin thinking about what results output you want to collect for the benchmark, you should also start thinking about how you will correlate data from different tools that all pertain to the same test runs, and how or where you will store the data and archive it.

Please refer to 2.6, “Organizing Results” on page 20 for a suggestion of how to organize your output data.

Even though you may not know all your requirements at the detailed level now, you should begin identifying them as they become known.

5.4.3 Summary

As stated in this chapter, the main deliverables of the definition phase are:

- A comprehensive set of *objective statements*.
- *Success criteria* defining what will constitute a successful effort.
- A complete *workload definition* which is a reasonable subset of the customer's total set of jobstreams.
- *Measurement criteria* and a well thought out plan for collecting, archiving, retrieving, and reporting on the benchmark results using correct metrics.
- Initial *hardware, software,* and *skills requirements* for the project.

These deliverables do not necessarily have to be complete before continuing to the next phase of the project (the *planning phase*), however, we recommend that they be as complete as possible before entering the planning phase. They should at least be in draft mode, and be complete enough to complete the *project planning session* that will be conducted in the next phase.

Chapter 6. The Planning Phase

The *formal* beginning of the *planning phase* of the benchmark project begins after the definition phase has begun and enough is known about customer desires and requirements to make intelligent decisions about how the project will proceed.

As stated previously, the items from the definition phase (found in the preceding chapter) do not have to be complete, however, if they are not, then time must be spent in the planning session addressing them, which will impact the productivity of the planning session. Therefore, every attempt should be made to have them completed. They *must* be finished before the end of the planning session because all subsequent activities depend on their completion.

The expected deliverables from the planning phase are:

- Completed objectives and success criteria statements
- Well defined requirements for:
  - Hardware
  - Software
  - Skills
- A defined workload that is to be benchmarked
- An understanding of how to build a half-day work plan
- A list of items requiring follow-up actions
- A tentative start date for the project

6.1 Planning Session

A planning session is one of the first activities that should occur for all the benchmark participants, as well as those who will do the planning. This meeting should be done as soon as practically possible, to provide as much lead time as possible for required work items. There will be many items identified that must be completed quickly so the benchmark can start at an early date. The planning session and the deliverables it produces will be the main items discussed in this chapter.

6.1.1 Attendees

Below are listed the possible roles that attendees at the planning session might have. There may or may not be a single person identified for each role. Likewise, people may share a role, or one person may assume multiple roles.

Management:

- Benchmark project manager
- Customer manager (person who approves the project)

Marketing:

- Vendor marketing or sales representatives

Technical:

- Vendor technical specialist
- Operating system specialist
- Performance specialist
- Application specialist
- Database specialist
- Customer programmer
- Consultants

6.1.2 Methodology

The planning session is most effective when conducted face-to-face. The benchmark project manager may schedule the session. He or she may act as session leader, or the vendor sales representative or vendor technical specialist might take this role. The session location may be the vendor's sales site, the customer location, or at the benchmark center. The location is usually dictated by where most of the people who need to attend the planning session reside, and having the other attendees go to that location.

Whoever takes the role of session leader should appoint an assistant who will keep minutes of the meeting. If possible, a large flip chart and stand can be used to list all ″To Do″ items that are brought forth. The session leader should pass around a roster requesting each person's name, phone number, e-mail address and role in the benchmark project. The session leader should prepare and distribute an e-mail within two working days, describing the highlights of the planning session and the agreed upon list of ″To Do″ items.

With current travel costs and increased demands on human resources in all aspects of business, it may not be practical for all participants in the planning session to meet in one location. An acceptable alternative is to hold the planning session through a telephone conference call. As in the face-to-face meeting format, the session leader who hosts the session and acts as discussion leader should request the names, phone numbers, e-mail addresses and roles of all attendees. The session leader again commits to write and distribute an e-mail note within two working days, listing the session highlights and ″To Do″ items.

It should be noted that the initial planning session meeting is a *starting point* for planning only. More follow-up meetings or conference calls will probably be required as planning progresses. If the project is complex and will require a long time for preparation, periodic status meetings or calls should be scheduled so that all participants are informed of the project progress.

6.1.3 Agenda

Below is a topical outline with discussion points that can be used as a meeting agenda. We will explain the agenda in this section, but you will find it in outline form in B.2, "Sample Benchmark Planning Meeting Agenda" on page 115. The sample agenda can be distributed as needed.

6.1.3.1 Account Background and Status or Customer's Business

Someone from the customer team should provide a brief overview for all participants about the customer's *business*, plus any pertinent background status, such as either recent or pending political changes, planned migrations of IT hardware or software, and so forth. The benefit of this activity is that it allows all participants to understand the customer environment, which helps them to understand why some non-technical issues may be very important to the customer.

6.1.3.2 Customer Objectives - In Customer's Own Words

The person on the customer team who is the main decision maker or manager should at this time either review the *benchmark objective statements* formulated earlier as discussed in Chapter 2, "Benchmark Basics" on page 7 and Chapter 5, "The Definition Phase" on page 47, or should at this time lead the discussion to complete the objective statements and acceptance criteria. Any benchmarking constraints such as an apples to apples comparison, particular required hardware or software, (such as SMP), should be discussed.

All subsequent planning activities should be conducted using these objectives as the primary guideline.

6.1.3.3 Success Criteria

The success criteria should already be familiar to all attendees based on the work done during the definition phase. It should be reviewed at this point to verify it and communicate it to all attendees. You should be aware that as the planning progresses, the benchmark objectives and success criteria may need to be adjusted, so it should be considered a *work in progress.*

6.1.3.4 Benchmark Workload Definition

Refer to Chapter 2, "Benchmark Basics" on page 7 for detailed explanations of the various types of workloads, such as batch, interactive, client/server, and so forth. If the workload has been defined adequately prior to the planning session, it should be reviewed briefly at this time for a final check of its validity. Keep in mind the following points as you either develop or review the workload to be benchmarked.

The workload should encompass the *intended real use* of the system. The jobs that are included should:

- *Be organized* in a logical manner, that is, if one job loads data that is later queried or manipulated, then this job must logically be executed before dependent jobs.
- *Be representative* of the total production environment. What this means is that when all the work is completed for defining the benchmark workload, it should be reviewed and the following questions asked about it:
  1. Can this workload be *implemented* for benchmark testing purposes without requiring an unmanageable work effort?
  2. How well does this workload *support the objective* statements we defined earlier? If it doesn't support those objectives it is not the proper workload.
  3. If we use this as our test workload and we are able to achieve the results stated in our acceptance criteria, are we confident that the actual production system we install based on this test will perform as we predict?
- *Be Ported and tested* with the data that will be used prior to the start of the benchmark.
- Have *variances defined* if there will there be any in the workload.
- Have the *test duration defined.* What will our *timing window* be?
- Have a *data reset* methodology established.

The *success criteria* should state clearly what results will be deemed satisfactory.

The *measurement metrics* that will be used must be established. (How and with what tools will the results be measured?)

Any *preset success bars* should be defined. Are there hard requirements to be met that might be different from additional acceptance criteria?

The planned *measurement tools* to be used must be defined. Do we have access to them? Is someone sufficiently skilled in their use? Will we have to develop any unique, specialized tools, such as hooks in our application code that will provide measurement data?

6.1.3.5 Benchmark Requirements

It is critical to the success of the project that all requirements in the following areas be clearly defined so that any resource voids can be addressed, proper cost and time estimating can be done, and plans can be made to acquire and assemble all resources at the proper time and place.

- Hardware requirements

  Data staging disks or database backups. Could they be the same disks or must separate disks (spindles) be used?

  Type and size of disks to be used.

  Requirements for spindles versus space.

  Mirroring disks.

  Archival disk storage (for results and testing scripts).

  Number and types of tape drives required (parallel loading and backup and restore tape requirements may require you to reserve some extra).

  CPUs, memory, disk, tape, network and any other miscellaneous I/O devices required for those systems listed below:

  – System under test.
  – Connected systems (performance monitor).
  – Remote terminal emulator systems.
  – Any required other equipment manufacturers devices.

- Software requirements including version levels and fix levels for the:

  – Operating system.
  – Licensed Program Products (LPPs).
  – Relational Database Management System (RDBMS).
  – Third-party copyrighted code.
  – Customer's own code.

- Locations of tables versus each table's respective index.

- Log files (keep each log file or any file sequentially written on a dedicated disk spindle for performance purposes).

- Make sure that all versions of the software fit and work together.

- Skill requirements (make sure all are available for the following areas):

  – Project management.

  - Benchmarking.
  - Systems administration.
  - Operating system tuning.
  - RDBMS DBA.

- Data Requirements:

  - Size and complexity.
  - Sources and conversions.
  - Data generation.
  - Media and tape formats.

- Resource availability:

  - Equipment
  - Software
  - Participants.

6.1.3.6 Benchmark Activity Workplan

At this point enough should be known about the project to enable the customer team to begin the development of a *benchmark activity workplan.*

The benchmark activity workplan is a documented work plan that can be a very productive tool during the *execution phase* of the benchmark. It documents, in one-half day increments, what tasks will be performed by what subteams or individuals for the entire proposed time spent at the benchmark center. A team leader should be identified as the focal point for this effort.

You might ask of what value this is. The value is that by performing the planning steps required to create the document, issues can be resolved in a *proactive* mode instead of a *reactive* mode such as:

- Which tasks, in what order, need to be performed from the time we walk into the benchmark center until we finally leave?

- How and when we will verify that the hardware and software we requested is properly installed and functional?

- The utility programs, third-party software and language compilers that should be on the system.

- Where can we insert decision points into the plan that will allow us to:

  - Check how well we are keeping to our plan.
  - Make decisions about unplanned events.
  - Invoke a contingency plan that will get us back on our schedule if required.
  - Back out some changes that were not productive.

  - Select alternatives to complete the benchmark ahead of schedule, get better results, and possibly generate better reports.

- How long we should allow for terminal emulation *script development* and *testing.*
- How we will insure that a task works on the *real* terminal before trying to capture a script for it.
- How long each *test window* should be.
- When we should collect and archive output results data.
- How often we will need to refresh our data.
- What is the most efficient methodology for doing our data refresh (such as, can we use staging disks and refresh from disk each time instead of rerunning the tape restores)?
- Will a more efficient data refresh scheme involve an increase in required resources? In other words, will we need more disks to use this methodology, or can we use a portion of our *test* disks that is not used during actual testing? This assumes that only a portion of the test disk is required for actual test data.
- Is there is a way that we can break the required work into parallel tasks that are executed independently of each other, then merge them back into the mainline effort, to save time and money?
- If there are activities that the ″yyy″ team can be doing while the ″xxx″ team is doing their work that will add efficiency to the project.
- How and when will we integrate the parallel work efforts back into the mainstream of the benchmark?
- The kinds of housekeeping activities we should plan to do after we finish actual testing, and how long they will take. For example, activities may include archiving test scripts and binaries that were changed during the testing, backing up data files, tools, and utilities to be used for future benchmarks and preparing reports.

Be Safe and Plan Ahead

If you *plan* and *prepare* to have a follow up test, it will more than likely *not* be required; if you plan to *not* return at a later date and *do not* preserve any of the current work effort, ″Murphy’s Law″ states that you *will surely* be back later.

The value of the benchmark activity workplan is that by developing it you can see:

- How much time the project should realistically take
- How the various tasks that make up the benchmark relate to one another
- What the task interdependencies are
- Which tasks can be done in parallel, thereby saving time and money
- How the schedule can be optimized
- How the work flows from start to finish

You should try to complete the benchmark activity workplan as soon as possible, so it can be reviewed by all participants. Their feedback often helps streamline the plan and results in time and cost savings.

Consider doing one or more of the following:

- Drop any non-critical tasks that will not improve the quality of the project.
- Think about how you can optimize the work flow based on the workplan.
- Can you get more equipment to use during the benchmark that will shorten the time required at the benchmark center?
- Will the increased cost of the additional equipment be offset by decreased personnel costs? In other words, is this a good financial trade-off?
- Are there people currently working on the planning or scheduled to work on the execution that could be used more productively elsewhere without impacting the quality of the project? Identifying problems here can save money.
- Are there some tasks currently planned to occur during the execution phase that could be done in the customer's shop during the planning phase, thereby shortening the time at the center? Consider items such as code and data conversion, script development and testing, and database table preparation if a subset or superset of the production database will be used.

Once the plan is completed, have it reviewed by someone not involved in its creation, then refine the plan based on feedback received.

Be sure to keep participants notified if the workplan causes changes such as:

- Who or when participants will be involved during execution.
- If resources such as equipment or skills must be rescheduled.

See B.3.3, "Sample Checklist for Benchmark Activity Workplan" on page 118 for a sample benchmark activities workplan.

6.1.3.7 Open To-Do Items

During the planning session a list of To-Do items should be kept, preferably on a large flip-chart or some other media that is visible to all participants during the face-to-face planning session. Having the list visible allows participants to review it during the day, which often prompts thinking about additional items.

For face-to-face as well as conference call planning sessions, the maintenance items listed below should be done at the end of the meeting, after all other business has been completed, but while all participants are still in attendance. This allows all interested people to be involved in addressing the issues and encourages better acceptance of the plan. Participants should be *encouraged* to add items to the to-do list during the planning session. Though it may seem like too many trivial items would be added, the real effect is that items that were overlooked are recognized, and problems are resolved in *planning mode* instead of during the execution phase. Any trivial items will be eliminated during the final step of the to-do item review and task assignment.

The To-Do list should be reviewed at the end of the planning session and the following steps should be taken:

1. Review each item and compare it to others to eliminate redundancy or the presence of items on the list that were resolved during the session.
2. Assign *one person* (only) to be responsible for completing the item or resolving the issue.
3. Go back and renumber the items after the list has been completely reviewed and reorganize the tasks chronologically.
4. If there are a large number of items or a rather long time span that the items cover, one or more *checkpoint calls* should be scheduled to review the status of all issues and items. Without fail, a last review should be conducted at least five working days prior to start of the actual benchmark execution.
5. The checkpoint calls can be used to insure that the planning work is on schedule, or if not, give advance warning to everyone that equipment and other resources may need to be rescheduled for a later time.

6.2 Summary

If the planning phase is done properly, it may be the most work-intensive step of the entire project. Many times participants see the intensity of the activities at this phase and are not convinced of their value. However, you should keep in mind that every unresolved to-do item could either add undue cost and complexity to the project, or worst case, cause it to fail.

The benchmark activities workplan can be the most effective planning tool for completely understanding the size, scope and flow of the benchmark project tasks.

Chapter 7. The Preparation Phase

Up to this point, all work has been more or less paperwork. Now we start to move real things around: hardware, software and even people. This phase of the project should be finished before the customer arrives at the benchmark site.

7.1 Installing Hardware

Hopefully, most of the hardware needed for the benchmark is already in place because it is never shuffled around. But sometimes we have to arrange for additional items to be delivered and installed.

- Make sure there is enough time left to allow for some delay in the delivery of the items. Do not assume everything will be there on time.
- Get information from the sender on how the items have been shipped so you can track the shipment in case of delay or loss.
- Get some information about the size and weight of the items. Make sure there is enough room to place them and enough room for installation and maintenance work. Do not assume everything will fit.
- Find out how the items will be connected to your existing environment. Get detailed descriptions about cables, connectors, and the like. Do not assume that all necessary items will be included in the main delivery.
- Sometimes there are different ways to connect items to your systems. Make sure you understand the differences and the implications of each method. Choose the one that is best suited for the purpose of the benchmark (this is usually not the cheapest one). Order additional cabling if required.
- If you are not familiar with the equipment, get support. This is usually a big time-saver. Do not assume that the equipment comes with a manual and the installation will be easy.

Most of the problems that arise with hardware items cannot be fixed quickly. Sometimes you even need replacement parts which might take some days to be delivered after you discover that the original item was defective. If this happens to resources that are critical to the project, this could lead to negotiations about modifying the original benchmark plan.

7.2 Installing Software

Usually, the first piece of software that gets installed onto the systems will be the operating system. Next come some additional features like device drivers, monitoring tools and the remote terminal emulation software. Sometimes we even have the chance to install part of the application programs at this stage.

- If possible, re-install the operating system from scratch, even if it is already installed. Make sure you know the exact software configuration. Do not assume that everything is already installed properly and no changes have been made that could influence the behavior of the system in a way you do not want them to.
- If the operating system consists of several modules or parts that can be installed optionally, decide which ones you want to use. Generally, it is better to install a bit more than you need than to forget an important piece.
- When installing the operating system, put it on disks of its own. If possible, do not put application code or data on the same disks. This helps to analyze the source of any disk bottlenecks later in the project.
- Since installing over the network is usually the fastest way, using another machine in the network as a repository for software images and installing from there saves time compared to installing from CD or tapes.
- If you use equipment you are not familiar with, find out about the software requirements for its operation. Do not assume that the equipment comes with all the software you will need.
- Get the latest bug fixes. Do not assume that a bug that you do not know about will not hurt you.
- Before installing any software you received from the customer, make sure the legal aspects of this transfer have been settled. Find out whether you need a licence key or some special hardware feature (such as a dongle) to install or operate the software on your system. Do not assume that the tape or CD you received contains everything you will need.

With access to qualified software support, problems in this area can usually be solved within a day or two. More severe incidents are typically due to errors made during the definition of the benchmark project, such as not identifying the correct software versions or not properly checking their compatibility to other programs or hardware used in the project.

7.3 Customizing the Basics of the System

After hardware and software are successfully installed, some parameters have to be adapted, such as hostnames, network addresses, or the number of users allowed on the system. Most of this will be platform-specific. On an AIX system, for example, you might also want to add more paging space, create printer queues or additional file systems and define some users and groups.

As always, it is a good idea to have a plan before starting to apply changes to the system. Therefore, we suggest that you first collect some information about what has to be adapted to be well-prepared for the execution of the benchmark tests. Also, do not forget to document all changes you have applied to the system.

7.3.1 Collecting Information

In most cases, the documentation for the operating system and applications you will use in the benchmark contain a chapter about performance issues and tuning. Use this documentation to find more details about which system parameters need to be adjusted.

For an AIX system, important information on various performance aspects can be found in the *AIX Performance Tuning Guide* (SC23-2365).

The information contained therein should be carefully reviewed, since some of the hints and tips included there can be applied at this stage of the project.

7.3.2 Using a Customization Script

If you find you would do similar things on many machines, think about writing a customization script. You can then keep this script after this project is over and reuse parts of it in projects to come. After doing the customization in several projects, you will eventually end up with a general purpose customization script that will do almost everything you might want to do at a particular stage of a project.

If there are items you have to chose a name for, such as network interfaces, database files, or dummy users, try to find a meaningful naming convention. If you have, for instance, a large cluster or an RS/6000 SP with many nodes each of them have several network interfaces, choose names that make the items easy to identify.

This also helps when creating scripts that handle these items, since you do not need to list each item individually (and maybe omit one), you can use expressions with wildcards that will automatically match to a subset of the names, or loops that generate the names automatically according to a specific algorithm.

Using a script also gives you the advantage of being able to easily keep track of what you did during the customization of the system. All actions can also be reproduced later, and the risk of overlooking any of the many tasks that have to be performed is reduced significantly.

7.3.3 Examples

In this section, we will discuss some of the parameters that most-likely need to be changed in an AIX environment.

This collection is far from complete, and maybe not all of the parameters mentioned here need to be changed for your project, but we believe this is a good starting point for further considerations.

7.3.3.1 System Parameters

After installation of the operating system, many system parameters will have their default values, which might not be appropriate for the project.

On an AIX system, the parameters most-likely to be changed include the following:

- *Number of licensed users:* Make sure you have enough user licenses for your project. Increasing this number requires a system reboot to become effective.
- *Maximum number of processes allowed per user:* Although this parameter does not apply to a root user, it is likely to be increased because many application processes are owned by a specific application user (the database, for example) and the default value of 40 is not sufficient in most cases. Setting this to 4000 is usually enough, although some applications may need even higher values.
- *Fast boot:* On some SMP machines (particularly when they have many processors and much memory) booting could take more than an hour to complete. Sometimes there is an option to select a fast boot, which will skip some of the most time consuming tests on restart. Since some system parameters do require a system reboot to become effective, you can save a lot of time by activating this option.

7.3.3.2 Paging Space

In AIX, paging space is part of the Virtual Memory Manager (VMM). Memory pages that do not have their place somewhere in the file system will be stored in paging space if they have to be removed from real memory.

As a general rule, you should have at least as much paging space as you have real memory. Instead of having one large paging space on a single disk, it is usually better to have up to four paging spaces of about the same size, each placed on a different disk. Having more than one paging space on the same disk is not recommended, since once the paging spaces are accessed, this disk is likely to become a performance bottleneck.

Be aware that some rules stating that your paging space should be a multiple of your real memory do not usually apply to systems with 2 GB of real memory or more.

7.3.3.3 Asynchronous I/O

Using Asynchronous I/O will usually improve your I/O throughput, especially when storing data in raw logical volumes (as opposed to file systems). The actual performance, however, depends on how many server processes are running that will handle the I/O requests.

A rule of thumb suggests to limit the number of servers to a maximum which is equal to ten times the number of disks that are to be used concurrently, but not more than 80. The minimum number of servers should be set to half the maximum number.

7.3.3.4 Network Parameters

In a client/server environment, or whenever the network is expected to be an essential part of the environment, review the network parameters.

Network options: Especially when using an RS/6000 SP with a switch, the default values of some of the network option parameters are much too low.

Keep in mind that changes using the `no` command are only temporary and will be lost after the next system boot, unless they are put into a script that is run on each system restart, usually /etc/rc.net.

RS/6000 SP switch parameters: In the RS/6000 SP, the switch uses memory buffers for TCP/IP communications. The size of these buffers should be increased.

NFS parameters: If the Network File System (NFS) will be used extensively during the execution of the benchmark, you might want to adapt the number of nfsd daemons on the NFS server and/or the number of biod daemons on the NFS client system. See the *AIX Performance Tuning Guide* (SC23-2365), Chapter 9, for some guidelines on which numbers would be appropriate.

7.3.3.5 User Parameters

If you need files bigger than 1 GB, the users need to have their file size limits adjusted. If many users are involved, consider changing the default value in /etc/security/limits.

7.4 Checking Some Basic Functions

It is always a good idea to check whether things are really working as expected. Before the customer arrives:

- Check the network connections and name resolutions (Are all nodes in the cluster known to each other and can they exchange information?)
- Check that the users can login (locally as well as remotely) and set their passwords to something that can be easily memorized.
- Check that all disks are accessible and record their numbers; this will help to identify them later when you start installing a database.
- Check that peripherals like tape drives or printers are operating properly.
- Check that any tapes you have received from the customer have the format that has been agreed upon and can be read on your system.
- Check that the monitoring tools you intend to use are installed correctly and give reasonable results when applied to some basic tests such as copying a file over the network or running a small program doing some calculations.
- Check whether all requisites for the application to be tested are installed.

Extensive checking is strongly recommended. This also helps to detect any errors that might have been made during the basic customization at a stage where you probably have some time to correct them carefully.

7.5 Documenting the Setup

At this point the environment is prepared as far as it could be, without direct participation of the customer. This is a good point to start some of the documentation work you will be doing throughout the execution of the benchmark by collecting the basic information about the setup. This includes the following:

- A complete list of hardware installed, including configuration details such as number of CPUs, amount of memory, adapter cards and other internal devices for each of the systems and peripherals.

- A complete list of software installed, including version numbers of operating system modules, device drivers, monitoring tools, remote terminal emulation software, and application code.

The easiest and also most reliable way of obtaining this information is to record the output of the respective operating system commands (for AIX this would be `lscfg` and `lslpp`, for example) and keep it on a diskette for later reference.

7.6 People's Needs

When the participants in the project arrive at the benchmark site, they will need some office space where they can sit and work on their tasks. They also need some basic materials to work with, so try to optimize the working environment.

- Make sure all participants find the way to the benchmark site. They should have the exact address, a map, written directions on how to get there and the name of their first contact.
- If possible, the participants should not sit in the machine room where the surroundings are often noisy and not very comfortable.
- The participants should have access to some basic system documentation, like operating system commands or programming language syntax. Make sure the manuals are available either in hardcopy or online.
- They might also want to have access to a meeting room, telephone lines and a fax machine, sometimes at short notice. It is good to make some reservations early.
- If you think of entertaining the customer's employees by providing free drinks and snacks or meals, you should not forget to order them in time.
- Have some marketing materials on hand. Some of the participants might not be familiar with the product line or the advantages of the products or services your company has to offer.

All this might not influence the results of the benchmark directly, but it will very much help the sentiment and attitude of the people involved and create a better working atmosphere.

7.7 If Something Goes Wrong

By taking care of the hints and tips provided above, you will be able to avoid many of the common pitfalls that can bring a benchmarking project to a sudden end even before the first measurements have been done.

Nevertheless, you will still encounter situations where something is not working the way you would like it to. Hopefully, your schedule allows for some time to correct this.

- As a technical person, if you encounter a problem, do not keep it a secret. Tell the project leader about it, since they might need some time to think about what to do if it can not be fixed in time. Do not forget to talk to him them in case the problem could be resolved.
- As the project leader, make sure that you keep up-to-date with the latest developments. Do not assume that no news is good news. The technical people are often quite busy, especially when dealing with problems, so they might not think about giving you all the details you might need.

Communication between the participants is a key to the success of a project that involves a team.

Chapter 8. The Execution Phase

At this point the hardware has been set up and the different software products have been installed. No customer-specific configuration has taken place. The execution phase typically starts when the customer arrives with the tapes containing their application code and data and their environment customizing scripts.

The execution phase is an iterative process where working hypotheses will be made and tested. Each run must be monitored and analyzed upon completion. New hypotheses will be made and acted upon prior to each new run. Documentation must be made to keep a history of all the runs and their outcome.

At the end of this phase, you will have obtained all the raw data needed to assess how successful the benchmark has been. Keeping in mind that a report will have to be written, you must keep all the necessary information pertaining to the history of the benchmark. This includes any scripts that were written, parameters that were used and options that may have been changed.

8.1 Introducing the Environment

The customers have just arrived at the benchmarking facility. Show them around the basic facilities of the building where they will be working (such as restrooms, emergency exits, coffee machines, meeting rooms) to help them become familiar with their new working environment.

You may not always be with your customers. Provide them with a backup name to use when you cannot be contacted. They should always be able to either find what they are looking for by themselves or contact someone who can help them. Typically, customers will require access to a telephone line, a fax machine and a printer. IBMers will also look for a PROFS/NOSS terminal or want to connect their laptops to the network and use their Lotus Notes accounts.

Give the customer all the necessary preliminary information concerning the benchmark environment. They should have a list of the defined user accounts and their associated passwords. This includes knowing where the machines are and how to access them, both physically and remotely. This means that they will have been provided with a description of the networks they may have to use (IP addresses and naming conventions).

If you provide this information in written documents, the reporting phase will be easier.

8.2 Verifying the Environment

Check that each machine has all the resources that have been agreed upon, number of processors, amount of memory and number of disks, both internal and external. Check that the operating system is at the proper level and that this is also true of all the software that is installed.

This task can be made much easier if the customer is presented with the scheduling request sheet that was defined at the end of the planning phase and compares it with both the hardware and software that have actually been installed. Any discrepancies should be identified and dealt with as quickly as possible.

Make sure that the benchmark activity workplan is reviewed before actually starting the benchmark. This will ensure that all participants have a common understanding of the tasks to be accomplished, of the responsibilities each one has and of the time constraints that must be respected.

In case the benchmark is commercially-oriented, the database that will have to be created will probably be at the core of all performance issues. Take some time to carefully review the database schema and its physical implementation on the tested configuration.

8.3 Customizing the Environment

This part deals with the installation of the actual benchmark code to run and its associated data, as well as creating the customer-specific environment required to run the benchmark.

8.3.1 Using a Staging Area

Code, data and scripts usually come on tape. The first task will generally be to read the tapes.

Reading the customer's tapes can be a rather lengthy operation. You may need to read the tapes more than once. Of course this could be due to disk problems, but it could also simply be because the database needs to be reloaded prior to each new run. Using a disk as a staging area, writing everything there and using this disk to do the proper installation might then be a good idea. Just think how much time will be saved reading from disk as compared to reading from tape again, in case re-installation or reloading is necessary.

Create the staging area and write the tapes to it. The basic rule would be to use as few disks as possible to have enough disk space to accommodate what is on the tape. However, there can be very good reasons to not follow those guidelines. The number of disks to use for the staging area can depend on how important performance is. Will the data to be loaded be read directly from the staging area, and is the loading part of the benchmark? From how many files will you load? Can those files be loaded in parallel? The answer to those questions will indicate how many disks you should include in your staging area volume group and to which physical disks each file should go.

Some benchmarks may require the same tests to be run on different machines. However, it is not always possible for many machines to coexist and share the same staging area. This means that the same code and the same set of data may have to be similarly installed as many times as there are different machines. This does not necessarily mean that the staging area should each time have to be re-created. You should try and make it as easy as possible to use the same data and disk on all machines.

Under AIX, this can be achieved by creating the staging area on one or more external disk volume groups. Those disks can be unplugged from one machine and later reconnected to another. Using the `importvg` command they can then be made available on the new machine. Repeating the operation on each machine, you can make your staging area locally available on all the machines that participate in your benchmark.

8.3.2 Creating Groups and User Accounts

To create the necessary users you will need to have a list of those users and the groups to which they belong. Users and groups may also need to have specific IDs, home directories or authorizations. Make sure that this information has been provided.

In a cluster of machines, users often need to be known across the cluster. Those users can either be created on one of the machines and made known to the others through a network or created on each machine. However, the result is not identical. In the first case there is a single user with a single home directory shared by all the machines. In the other case, there are as many copies of the user as there are machines in the cluster, each copy having its own local home directory. What this means is that, in the first case, a file written in the home directory of a user will be immediately accessible from all machines, while in the second case it will have to be copied to each local home directory before being accessible from all the machines in the cluster. There are advantages and disadvantages to both solutions.

The first solution only requires users to be created once. There is no consistency issue. Under AIX, they will be made known to the other machines in the cluster through NFS mounts. The drawback is of course that it introduces a single point of failure. Furthermore, NFS reads and writes slowly compared to local I/Os. Could this create a performance bottleneck or significant slowdown?

If the answer is yes, you might need to use the second solution and create all your users locally on each node. You should then explicitly specify a value for all of the parameters that are associated with the group or user account created. One of the drawbacks to this method is that consistency across nodes will have to be verified and managed manually.

There is a way of creating users that is specific to the SP and will ensure that those users are available across all the nodes of the SP. An SP user's home directory resides on only one node. If you create all your users' home directories on the control workstation, you should be aware that there will be no way for you to NFS mount those directories using a high performance network such as the switch. You should also be aware of the fact that those users will no longer be available in case the control workstation goes down.

Though it might not be as important with creating users as with other tasks, try and always use scripts. This allows easy repetition of a task when necessary, and helps facilitate documentation.

Note

Avoid using all-purpose and all-privileged user accounts. Under Unix, this means that you should not use the root user account unless you have to.

8.3.3 Creating File Systems

The directories where the code is to be installed is usually customer-imposed but the underlying DASD structure usually is not. In this section, we will discuss how the specific files and directories will be mapped onto the physical disks.

Should we create one big file system and install everything there? The advantage of such a solution is that, as long as we make sure that some space is available in that file system, we do not have to worry about how much space each individual component requires. On the other hand, it also means that it will be difficult, if not impossible, to isolate one component from the other. And that means that little I/O tuning will be possible. That, in itself, is not a problem if your benchmark does few reads and writes and is mostly CPU-intensive. It might even make your life much easier.

On the other hand, it will be a serious drawback if your benchmark is I/O intensive. In that case, you need to have better control over how your I/Os can be spread or rebalanced across your physical disks. Create specific logical volumes (or volume groups) to host specific parts of your applications, keeping in mind the expected I/O activity. That will direct you as to the number of disks that each particular component should use and whether it could benefit from striping or not.

Experience shows that accessing code for execution usually is not a performance bottleneck and that there usually is no need to spread the code across more than one disk. Once again, you should check that it is the case in your particular benchmark.

8.3.4 Compiling Programs

There are a few questions that should be answered if your benchmark uses compiled programs.

Note

At this stage, programs should have been ported and tested, if necessary. Check that this has been done before attempting to compile. If it is not the case, the effort needed for such a task must have been estimated during the planning phase and appropriate skills must have been committed to the porting.

Are we allowed source code modifications and what improvement in performance may we expect from code optimization? You should be aware that code modification is a lengthy process that requires a good understanding of the application. The answer to this question will give you hints as to how much time you should devote to optimizing the source code.

Are you allowed to change the compiling options? This is usually a rather simple way of making sure that programs run better on the platform you are using. In case you are not allowed to modify any compilation options, you should always determine which platform will benefit the most from things as they are.

It may happen that the preprocessing of the programs that make up the benchmark requires one or more tables in a database to already be accessible. This is true with programs that include SQL statements and need to be pre-compiled.

8.3.5 Creating the Database

Commercially-oriented benchmarks will require a database creation and the loading of data into the created database. This phase will often be the basis of all subsequent performance issues.

As always, you should ask yourself what can be changed, what cannot, why, and who benefits from parameters as they are? How will the database be created? Have scripts been provided? If so, it might be a good idea to review them for obvious errors and then use them as a starting point.

8.3.5.1 Database Creation Parameters

You must be aware that some of the parameters used at database creation can only be changed through the database destruction and re-creation. They are specified at database creation time but may need to be repeated in the initialization parameter file. Particular attention must be devoted to those parameters.

8.3.5.2 Initialization Parameters

Initialization parameters are read at database startup. This is where most of the database performance tuning will take place. Most of the parameters in the initialization file can be changed. They require the database to be shut down and restarted. Start by choosing high values. Those will of course depend on your application.

8.3.5.3 Data and Index Tablespaces

This phase introduces the problem of data placement and I/O contention. As a general rule, you should try and isolate each component (that is, tablespace and table) as much as possible. The idea behind this is to identify as easily as possible any potential problem. If there are too many tables in a tablespace you might have difficulties relating an I/O contention with the use of a particular table.

The system tablespace should never include data tables. Be aware that unless you explicitly specify a tablespace for your table, it will usually be created in the system tablespace.

Should you create your database on file systems or raw devices? The answer might depend on how important performance is versus ease of use. Raw devices perform better than file systems, but are harder to maintain or back up. The answer might also be dictated by the functions or products that need to be implemented. Some products will require the use of raw devices whereas others, not being able to deal with raw devices, impose the use of file systems. Check each product for its prerequisites.

Which tables will never be accessed concurrently? These may reside on a common disk without creating any further I/O contention. On the other hand, a table and its associated index will be accessed concurrently. They should therefore always reside on different physical disks.

How many concurrent access will there be to a particular table? An OLTP oriented benchmark with numerous users concurrently accessing the same table will probably show an I/O contention that may be reduced by striping the tablespace containing that particular table over more than one physical disk. Mirroring will also have a similar effect, though it should, of course, not be implemented for that sole purpose. Striping and mirroring are sometimes mutually exclusive.

8.3.5.4 Rollback Tablespaces and Log Files

The default location of log files is usually inappropriate. They should be written to their own file system on dedicated disks. This is particularly true in OLTP-type benchmarks where much inserting, deleting or updating is done.

Create your log file system with enough space to allow for growth. Unless some archiving or pruning of logs is done, you might fill-up the log file system and prevent further database activity.

8.3.6 Loading the Data

The loading of the data is usually timed in a database-oriented benchmark.

Note

For performance reasons, indexes should not be created prior to loading the data. If the index already exists, it will be automatically updated each time a row is inserted into the table. It is usually much more efficient to drop the index, load the data and only re-create the index once the table has been loaded. Check that you have no indexes at this stage.

There are a few questions you should ask yourselves to optimize loading.

- Which method and tool are you using? Most database managers provide two methods to load a table from a file. The import method generally inserts rows one by one, logging each insertion. This method, though very safe, is very slow and should be avoided when massive loading is done. The fast load methods inserts the data directly onto the database disks without going through the database engine, and no logging is done. This method is of course much faster than the first one. Which method are you using? Is there any valid reason for using the slow import method?

The fast loading method usually requires loading configuration files. Check that you have them or, at a minimum, that you have all the information needed to re-create them.

- Do you have to time the loading of the whole database? What most customers are really looking for is the number of rows that can be loaded per second. This can be achieved by loading a subset of the database. More loading tests can then be done in the same amount of time. Of course the whole database will still have to be created at some point, but at that point you can use the method described in 8.4, "Backing Up the Database and Restoring It" on page 81.
- Even if you only have one input data file to read from, you may want to spread it over more than one disk to benefit from the striping and read ahead algorithms provided by the operating system. Make sure that no other activity is simultaneously taking place on these disks.
- Can your database manager accept multiple parallel simultaneous loads? If this is the case and data has been provided in a single file, then you should check why this is the case. An important point may have been missed during the definition or planning phases. Are you allowed to divide your input data file into parts and is it feasible? Are there prerequisites to parallel loading and have they been met on your system?

  If you have more than one input data file and you can do parallel loads, then you probably want to spread each file over an independent set of disks so as to reduce I/O contention during loads.

8.3.7 Creating Indexes

Check that the scripts for index creation are available or, should no such scripts exist, that all the necessary information has been provided for you to create them.

Have the indexes been appropriately designed? An index created on a table using a column with a cardinality of two will probably be of little help. Check that the indexes that need to be created use columns with high cardinality. Also check that those columns will be used in the queries that will be run and that they make appropriate use of any defined primary and foreign keys.

8.3.8 Verifying the Database

Is there an easy way to verify that this phase has been correctly accomplished? If a reference query or set of queries has been defined you should run it now and check that the result output corresponds to that obtained on the original system.

You should also check, if necessary, that your client/server connection is operational and, typically, that you can query the database on the server from the client.

8.4 Backing Up the Database and Restoring It

Once your database is set up and your initial data is loaded, you might consider backing everything up. Indeed, a run might change the data in your database making it an unsuitable starting point for subsequent runs. Unless you have prepared for a fast method of returning your database to its initial state, then you might have to go through the lengthy process of deleting all the data in your database and reloading it from the original files prior to each new run.

If a small volume of changes has been made to your database and if those changes can be precisely identified, a script can be created to return your database to its original state. The rows that have been deleted can be inserted again, and those that were updated can be returned to their original values. Check that this operation does not create too much fragmentation in your tablespaces.

However, this solution cannot always be used, either because the changes are too numerous or because they cannot be precisely identified. You might have to resort to operating system commands such as `tar` or `dd` to backup all the data, log and control files and allow for future quick re-creation of your fully-loaded database. You must shut down your database before the backup to make sure that you are backing up a stable environment. Make sure that your backup includes all files that make up your database. A single file missing could prevent you from restoring a viable environment. This method is not suitable for a production environment!

Instead of using a single command to back everything up, it is sometimes possible to divide your load into subsets that can be backed up in parallel. Make sure that each subset is on a different set of disks from the others and that each backup is written to its own disks or tape drive. This will reduce your backup time and your restore time.

Use a script to verify that your database has been correctly restored prior to each run. This will ensure that no table or index has been omitted. You can even include row or index verification in your script.

8.5 Customizing Additional Software

Many benchmarks require more than just system and database installation and customizing. As an example, you might need to install a transaction monitor such as CICS or Tuxedo, or include high availability with HACMP for AIX. These usually require very specific skills so that it is not unusual to end up having many people work simultaneously on the same system. Unless proper communication has been established between the different participants, you may find that what has been done by one will be undone by another.

Each participant should be assigned a specific user account to work with to reduce the risk of interacting inadvertently with the others. There will be times when the root user account must be used. Document very precisely what has been done under root and make sure that everybody else involved in the benchmark is aware of it.

8.6 Allowing for Result Collection

The pertinence of your run analysis depends on the data you collect during the run. Particular care must be taken with the result collection and analysis.

8.6.1 Using Monitoring Tools

What monitoring tools will be used? Check that they are available on your system. You should, of course, check that they will enable you to effectively gather data that can establish whether the success criteria have been met.

Will the tools be used through scripts that were provided? If not, check that you have all the information needed to create the scripts and obtain the expected results.

Have the selected tools already been tested on the selected platform? This may prove particularly important when running a benchmark on an IBM RS/6000 SP or SMP.

Some of the performance monitoring tools may need to be customized. Performance Toolbox for AIX (PTX) is one of them. An appropriate customization will enable you to collect pertinent results to facilitate analysis. More information on PTX can be found in *Customizing*

Performance Toolbox and Performance Toolbox Parallel Extensions for AIX, SG24-2011.

Verify that the tools you have selected do not create a significant overhead on the system under test. This would be the case if you were to display the PTX GUI on the system under test. Use the system that has been planned for this purpose during the definition and planning phases.

8.6.2 Choosing Granularity

What granularity should be chosen? If the run is supposed to take 24 hours, then it might not be a good idea to take measurements every second. You would probably be overwhelmed by the data generated. Keep the captured data to a reasonable amount. Remember that it will have to be analyzed.

Be aware, however, that choosing too great an interval could cause you to miss valuable information. An interesting way of dealing with this problem could be to simultaneously use different granularities. As an example, you might want to start a first `iostat` taking measurements every five minutes and a second one taking measurements every 30 seconds for five minutes every half an hour. Such a methodology will enable you to get both a general and a specific view of your performance during a run.

Allow for enough disk space for gathering results. This will depend on the tools' output and granularity, as well as on how much history should be kept. Do you want to keep the results of all the runs, only the last one or only the last five? A simple computation will then show you how much disk space should be devoted to that purpose. Keep in mind that some of the data will be gathered to understand how the application can be better tuned while some will be needed for the reporting phase. Those are often two different sets of data.

8.6.3 Analyzing the Data

To get a full understanding of the way your application works, you will often need to compare the data collected from different runs or, within the same run, relate data gathered with one tool with that generated by another tool. Be sure to make sufficient and efficient use of timestamps. You should also be able to relate data indicating a problem to the task that was being done at the time, and that is not always easy.

Do one test run and run your monitoring tools. Do the scripts work and can you make any sense of the data that you collected? This will probably help you make sure you are collecting the right set of data.

8.6.4 Selecting a Monitoring Interval

This phase is particularly important in case of an OLTP benchmark. As we mentioned it in 2.3.5, “Remote Terminal Emulation” on page 15, the emulated users will not all be started at the same time. Though the ramp up is often of interest and should be monitored live, the representative interval will only start when all the users have been started. It will end when the first user finishes and exits. More information on this topic can be found in 2.3.5, “Remote Terminal Emulation” on page 15.

8.6.5 Organizing the Results

Properly organizing your results archive is the only way to not overwrite any data and be able to relate results to a particular run. Refer to Chapter 2, “Benchmark Basics” on page 7 for a suggestion on how to safely organize your results.

8.7 Integrating the Customer Workload

Though the workload has been defined in the definition and planning phases, the scripts may not yet have been written or completely tested on the target configuration.

Always test a task live before scripting it. It is important to make sure that each individual task can be properly executed. Any error at this stage might be very difficult to identify once everything has been scripted.

Now that each script has been written, they should be tested individually. Once this has been done, you can start testing them as a group, first with a single user, then gradually increasing the number of users until you reach your final objective. The number of tests this operation will require will depend on much time you have planned for this phase and how long it takes to run one test.

If you want to avoid wasting precious time, you should make sure that an emergency exit has been provided. That exit would allow you to stop a test run before it naturally finishes. This could happen for various reasons, one of them being that you forgot to change a parameter, making the test pointless. Emergency exits are usually provided in the emulation packages used for OLTP benchmarks, but they are unusual with a batch benchmark. One way of dealing with that particular issue could be to use a specific user account to start the test. If the test needs to be stopped you can remove all the processes that belong to this specific user. Make sure this does not leave you with a half-cleaned environment. This is always a very unorthodox way of doing things but it often is the quickest, and speed is exactly what we are after.

Your scripts should allow you to know how many users are active at any particular point in time. How else would you be able to identify a significant monitoring interval? They should also warn you if users exit with errors.

Keep in mind that no matter what the nature of the workload, it should respect some basic rules:

- Representative

  This can only be the customer's responsibility since only they can have full knowledge of their working environment. How is each individual script designed? Has the customer taken an active part in that process? Have they reviewed each script and are the scripts representative of the work being done in the company? Have they reviewed how the various scripts will be combined to create the final and intermediary workloads?

  Is the test network used in a way that is representative of what can be observed at the customer site?

- Repeatable

  Have the scripts been run more than once to verify that the same scripts will always give the same results or at least results of the same magnitude?

  Some tasks have to be done before the scripts can be run again. Have they been clearly identified? They should have been tested and included in script files.

- Unobtrusive

  Is the driver system, as described in 2.3.5, "Remote Terminal Emulation" on page 15, incurring any overhead on the system being tested? You should also check that the driver system has been correctly sized for the tasks that have been assigned to it. An over-burdened driver system may impact how tests are started on the system under test.

8.8 Running the Benchmark

At this point you should have functionally tested the environment and been able to verify that each component has been correctly installed.

You are now ready to start running the benchmark and use your monitoring tools for online and future analysis.

The following sections focus on systems running the AIX operating system.

8.8.1 Analyzing Results

This section intends to give you some hints about the different types of bottlenecks you may run into. You should refer to *AIX Performance Monitoring Tuning Guide*, SC23-2365 for more information on performance issues.

8.8.1.1 Memory-Bound Systems

This typically means that you could identify a significant paging activity. This will be shown in the `vmstat` output. Occasional paging is not in itself a problem and might even be expected. In some cases, it might be better to have some paging occur than increase the amount of memory available on your machine, thereby increasing the price of the configuration you are testing. The priorities defined in Chapter 5, “The Definition Phase” on page 47, detail what may or may not be done.

8.8.1.2 I/O-Bound Systems

Though the `iowait` percentage shown in the `vmstat` output is often used to determine whether you are encountering an I/O bottleneck, it should nevertheless be used with caution, especially if the benchmark is running on an SMP environment. This is further explained in *RS/6000 Performance Tools in Focus*, SG24-4989.

On a uniprocessor, a high `iowait` percentage associated to a high `busy` percentage on a few selected disks usually indicates unbalanced I/Os that could impact performance. Use utilities such as `filemon` or `fileplace` to identify the files involved in those I/O operations.

If in spite of the high `iowait` percentage there is little disk activity, then you should check to see if there is simultaneous read and write activity on the same disks.

Be aware that iowaits are sometimes unavoidable. You may have to read huge files before any processing can take place. You could try spreading those files across more disks, though you would still have to read them before doing anything else. The question you should ask yourself is whether the performance improvement will really be worth the cost in disk.

8.8.1.3 Network-Bound Systems

Utilities such as `netstat` will show you how much activity is on your network interface. A high network activity, combined with a high `idle` percentage as reported by `vmstat`, will typically identify a network-bound system.

8.8.1.4 CPU-Bound Systems

This typically means that the CPU utilization and addition of the `user` and `system` percentages as output by `vmstat` tends towards 100 percent.

This is what we are usually trying to achieve. It usually means that none of the previously-mentioned bottlenecks have been reached and that we are making the best possible use of the most expensive component in a computer, the processing unit. The application, as it now is, runs as smoothly as possible on the present configuration. But it does not always mean that no further tuning should be done. The application itself may need to be optimized. How can we find out? We can compare our results with reference results obtained on the original platform. Does it show any unexplained discrepancies?

If you are running on an SMP machine, you should be aware that the percentages given by the `vmstat` command apply to the global CPU power of the machine. Having one CPU fully-utilized while the three others are totally idle will show your system as using 25 percent of your CPU, and you would definitely have a CPU bottleneck. If there is no easy way of correcting this, it may mean that the application cannot be properly optimized on an SMP. This should have been identified during the definition and planning phase.

We should also have a look at how the CPU utilization balances out between the `system` and `user` percentages. A high percentage of `system` (over 30 percent) should definitely be investigated. Keep in mind, though, that the `system` activity is application-dependent and that no value should necessarily be considered right or wrong.

What `runqueue` value do you have? A high `runqueue` value probably indicates that a lot of context switching is going on. This may explain a high `system` activity. Keep in mind that a `runqueue` value of 6 can indicate a problem on a uniprocessor and be optimum for a 6-way SMP.

8.8.2 Tuning

Due to time pressure, it's often very tempting to change more than one parameter at a time. Be aware that it is then often impossible to decide the relative importance of each of the changed parameters. Only change one parameter at a time. Each modification should be tested and its impact assessed.

8.8.2.1 Memory Bound Systems

Increasing memory is just one way of dealing with your bottleneck, so also investigate the possibility of rebalancing the amount of memory used by the various processes. Using tools such as `svmon` will enable you to choose the best option.

8.8.2.2 I/O Bound Systems

Moving the files that you have identified in 8.8.1.2, “I/O-Bound Systems” on page 86, from hot disks to less-utilized disks could reduce contention by achieving a better balance of your I/Os across the existing disks. You should also consider spreading some of your I/Os over a greater number of spindles and adding more disks to your configuration. Note that in this case what counts is the number of spindles that can share your I/O load. This would typically be the case when a table in a database is simultaneously accessed by a great number of users, each competing against each other to access the part of the table in which they are interested. Striping that particular table over a greater number of disks will reduce that contention and improve performance.

Frequent high `iowait` percentage on a particular set of disks means that too many disk reads or writes are being done simultaneously on the same disks.

8.8.2.3 Network Bound Systems

You should investigate your network throughput and find out whether you are approaching your type of network’s nominal throughput value. If you are, then you might consider moving to a better-performing network. If not, then you may need to further tune your network. The priorities defined in Chapter 5, “The Definition Phase” on page 47, direct you as to what may or may not be done.

You can verify your network interface parameters using the `lsattr` command. On the RS/6000 SP, the default setting for the switch’s memory buffers used during TCP/IP communication should often be increased. This can be achieved by modifying the `rpoolsize` and `spoolsize` parameters.

In case of high NFS activity, you may need to increase the number of biod and nfsd servers.

8.8.2.4 CPU Bound Systems

Is any application tuning possible? Provided the customer allows it, we should try and tune the application itself. This typically means analyzing and tuning the customer queries and/or optimizing their programs.

Query Analysis and Tuning: It is usually possible to generate a query access plan to verify how your database optimizer is actually handling a query. You should make extensive use of that possibility, especially if you can identify that one or more queries run significantly slower than expected.

The access plan will enable you to verify that your indexes are actually being used. It will also allow you to check how memory is being handled by your database. You might thereby emphasize a high sorting activity and realize that performance could be boosted if more database sorting space was used.

Remember that locked database resources are also a frequent performance inhibitor, especially in OLTP-type benchmarks.

Program Profiling: Tools such as `tprof`, `pv` or `trace` might allow you to emphasize problems within the programs themselves. Refer to *RS/6000 Performance Tools in Focus*, SG24-4989 for more information.

It might also be possible to rebalance some of the other resources. You may try to change the amount of memory used by the database manager even if this means reducing how much is used by the other applications.

Optimizing caching or changing scheduling priorities are also ways of possibly overcoming a CPU bottleneck. If you consider such tuning, remember to always start out by using high-level commands before using low- level, complicated ones. As an example, if you intend to change scheduling priorities, you should start with the `nice` command before using `schedtune`.

The last method to overcome the bottleneck would be to increase the CPU power of your machine. Has this been planned for? How difficult is it to add CPU power? If you are running on an SMP then it might be as easy as plugging in an extra CPU adapter, provided you can get one. On a multiprocessor such as the RS/6000 SP it might mean adding an extra node, but it also probably means redistributing the data. Can this be achieved in time?

8.8.3 Re-Initializing the Environment

Remember that you will often need to re-initialize your environment before starting a new run. Be sure to move all the results that you have collected during the preceding run to their backup location. You certainly do not want to lose any data.

8.9 One More Run

One of the issues that you will eventually come up with is deciding when to stop. It might always be possible to optimize further and get slightly better results, but will it be worth it?

8.9.1 CPU Bound

This can be either the best or the worst case but it might mean that you have the best possible results with this configuration. This has been addressed in 8.8.2.4, "CPU Bound Systems" on page 88.

8.9.2 Meeting the Success Criteria

We might argue whether meeting the success criteria really is an indication of when to stop. If you are involved in a competitive benchmark, then the real success criteria, in spite of everything that has been previously said, would be to do better than the competition. You may still decide to continue if major improvement in performance can be achieved with little effort.

8.9.3 Minimal Improvement Requiring Maximum Efforts

How much performance gain may we reasonably expect from further tuning, and how costly will that be? Would what may appear as a minor performance improvement make the difference between winning and losing? In that case, the word minor would be an erroneous qualifier. The risk must be assessed and responsibilities taken. There are no rules that can be strictly followed.

8.9.4 Running Out of Time

Even though this is never a satisfactory reason, it is nonetheless one of the most common reasons why we might put an end to our efforts. But isn't reducing that risk the goal of careful planning?

8.10 Wrapping Up the Execution Phase

You have now obtained the results that you were striving for. However, the execution phase is not quite finished. Copyrighted software has been installed on the tested configuration. You need to make sure it is deleted from the machines. In most cases, the codes that have been lent by third-party vendors must be returned. Action must also be taken to protect the code and data provided by the customer.

Only then can the configuration be dismantled.

8.10.1 Taking Backups

What if some time later the customer wants to reproduce the same benchmark on a new, more powerful machine? It would be a shame to have to start all over again, and very expensive! You should plan for the worst and back up the entire configuration.

It might be a good idea to divide your backup load into different types, each type being backed up on its own specific tapes. This method will minimize the impact of the loss of a particular tape. We should always plan for the worst.

8.10.1.1 System Backups

Take operating system backups of all the machines that were involved in the benchmark.

Things might be slightly different on an RS/6000 SP. In most cases, each individual node is not unique in its system configuration. Groups can usually be made of similarly-configured nodes. Only back up one node in each group. This will usually insure, if not a fully-automatic restore, a restore that requires minimal intervention. Do not forget to back up the control workstation!

8.10.1.2 Third-Party Vendor Applications

As stated earlier, you cannot allow anyone to use third-party applications outside of the scope of the benchmark for which they have been lent. The vendor applications must therefore not be backed up.

This does not include parameter files that may have been customized especially for the benchmark. Make sure that you identify those files and back them up.

8.10.1.3 Customer Code and Data

Customer code and data must also be backed up. Ask the owner if you can keep this backup or a copy of it. Remember that there are usually property and confidentiality issues. If you do keep a copy of your customer's code and data, be sure to document it in the final report, indicating who, at the customer's site, gave approval and mention any conditions that were specified.

If you do not obtain the customer's agreement to keep a copy of this backup, make sure that they understand that it will then be their responsibility to keep a copy.

8.10.1.4 Result Backups

Be sure to back up the raw results that have been collected during the runs. This backup should be kept on media separate from any of the other backups, to facilitate quick retrieval.

Note

Always verify that your backups have been properly written to tape.

Note

Always do multiple copies of the most important backups.

8.10.2 Removing Applications and Data

Customers usually want to make sure that their data and code have been removed from every machine before they leave.

Only now that all the backups have been done should you remove the applications. You may do this by actually deleting the files themselves. You may also unmount the file systems on which the files were located and remove the file systems. Both solutions guarantee that access to confidential or copyrighted applications or data is no longer possible.

Chapter 9. The Reporting Phase

This phase is rarely given the importance it deserves. It is not unusual to see people doing benchmarks only note down, just before they leave, how long each transaction took and imagine that it will be enough. We tend to forget that, as the saying goes, the work is not finished until the paper work is done. The report is the only document that will remain once the machine has been dismantled. It must be thought of as a document for future reference. It should therefore not only give figures but also describe a context and include comments and explanations. The raw results that have been collected so far will hardly be enough. They should be reorganized and emphasized to direct the customer, as much as possible, to making a decision that will be favorable to you.

Should the customer be the only target of the report? Wouldn't the other participants also benefit from being reported to? Not only would it make them feel more involved in the benchmarking process but it would also allow each participant to keep track of the successes and areas for improvement that were a part of the operation. We should remember that those people will in most cases take part in future benchmarks, and very often be in charge of setting up the production system.

The reporting phase is one that must be thought of at a very early stage in the process. Most of its different components will have been already created during the previous phases. Reporting will then be a matter of reorganizing all this information to give it some meaning. It always has a sales purpose. The account marketing or sales representative should therefore take responsibility for writing the report, as stated in the definition phase.

The reporting phase will end once the customer is handed the benchmark report and is presented with the results.

The following is a suggested outline of the parts that should be present in a benchmark report.

9.1 Management Overview

This part must be seen as an introduction and as a summary of the benchmark that was undertaken. We should keep in mind that very few people, if any, will actually read the report from beginning to end. Most will skim through and we must make sure that, even then, they understand the basics of both the purpose and the outcome of what has been done. It should therefore be kept quite general, not going too deeply into the technical matter yet emphasizing the major points to be carried across.

- The customer

  State who the customer is and give a list of marketing and technical representatives. Make sure all names are spelled correctly.
- The benchmark goal

  State the goal of the benchmark as it has been approved by the customer, which individual gave their approval on behalf of the customer company and when it was given.
- The success criteria

  Indicate the different success criteria that have been defined and approved by both parties. If there is more than one criterion, indicate the priority each one was given.
- The duration

  When did the benchmark start and when was it supposed to end. Was that achieved? If not, what reasons can be given?
- The constraints

  What were there constraints that had to be respected? How did they impact the benchmark?
- The participants and their responsibilities

  Give a list of all those who took part in the benchmark and what their jobs and responsibilities were. You should mention whether they are vendor employees, business partners, third-party or others.
- The outcome

  Give a summarized version of the results, stressing those that may put you in a favorable light. Have the objectives been met partly or completely? The results should always be connected to the success criteria. Are there results beyond the customer's goal? Those should of course be highlighted here.

Remember that this is probably the part that will be read the most. Its commercial implications should not be underestimated.

9.2 Hardware Configuration

Now that a general understanding of the benchmark has been established, some of the more technically-oriented people will want to know more about the physical implementation of the benchmark. Once again, you should not expect everybody to go through whole pages of `lscfg` output. Start out by giving a general idea of what hardware has been used and then go into the details, if interest is shown. This can be best achieved by giving a graphical representation of the configuration that was used, followed by a more in-depth technical description.

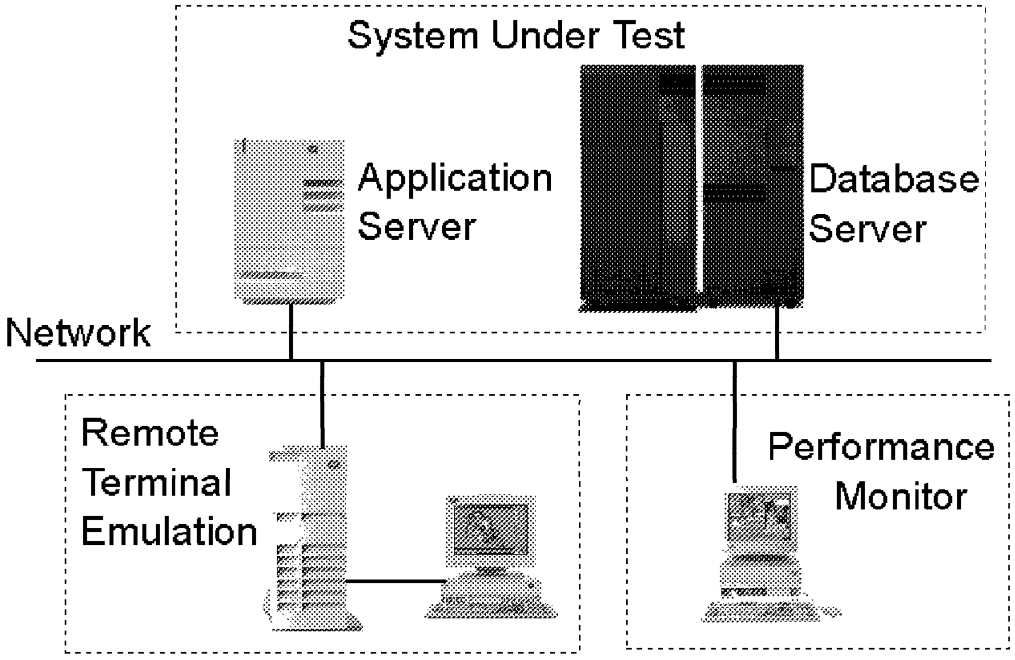


Figure 4. Graphical Hardware Configuration Overview

Upon completion of Chapter 7, "The Preparation Phase" on page 65, you probably used a command such as `lscfg` or `lsdev` to keep track of the hardware configurations of all the machines that were involved in the benchmark. They can help you go into more of the details of each configuration. Yet those files should be cleaned of all the unnecessary (less-important) information. How interesting is it to report that there were mice and keyboards on the various systems? They should also be reorganized to be more easily read. How much information is actually necessary has to be decided on a per-benchmark and per-customer basis.

Though only one system may have been tested, the configuration used may contain more than one system. This is typically the case in the example given in Figure 4. The driver system (RTE) should only be briefly described to allow the benchmark to be reproduced later if necessary.

Depending on how much information you think the customer will need, you may be as brief as the following example:

```
INSTALLED RESOURCE LIST FOR ”ALOHA” (SUT)
ALOHA is an 8-way R40 with 256 MB of memory, 3 x 4.5 GB Diff SCSI
and 3 x 2.2 GB Diff SCSI Disks.
```

If need be, and provided you have the information, you might relate the tested machine to some industry-standard benchmark figures:

```
258.0 SPECint_rate95, 200.0 SPECfp_rate95
244.0 SPECint_base_rate95, 189.0 SPECfp_base_rate95
```

It is doubtful that you will ever need to go as far as including information such as the following in your hardware report:

```
CPU + CACHE
+ proc0             00-0P-00-00       Processor
+ proc1             00-0P-00-01       Processor
+ L2cache0          00-0P-00-0L       L2 Cache
+ proc2             00-0Q-00-00       Processor
+ proc3             00-0Q-00-01       Processor
+ L2cache1          00-0Q-00-0L       L2 Cache
+ proc4             00-0R-00-00       Processor
+ proc5             00-0R-00-01       Processor
+ L2cache2          00-0R-00-0L       L2 Cache
+ proc6             00-0S-00-00       Processor
+ proc7             00-0S-00-01       Processor
+ L2cache3          00-0S-00-0L       L2 Cache
MEMORY
+ mem0              00-0A             128 MB Memory Card
+ mem1              00-0B             128 MB Memory Card
CONTROLLER ADAPTERS + INTERNAL DISKS
+ ascsi0            00-08             Wide SCSI I/O Controller Adapter
+ hdisk0            00-08-00-0,0      2.2 GB SCSI Disk Drive
CONTROLLER ADAPTERS + EXTERNAL DISKS
+ ascsi1            00-01             Wide SCSI I/O Controller Adapter
+ hdisk1            00-01-01-3,0      4.5 GB 16 Bit Differential SCSI Disk Drive
+ hdisk7            00-01-01-2,0      4.5 GB 16 Bit Differential SCSI Disk Drive
+ hdisk6            00-01-01-1,0      2.2 GB 16 Bit Differential SCSI Disk Drive

+ ascsi2            00-16             Wide SCSI I/O Controller Adapter
+ hdisk3            00-16-01-12,0     4.5 GB 16 Bit Differential SCSI Disk Drive
+ hdisk10           00-16-01-13,0     4.5 GB 16 Bit Differential SCSI Disk Drive
+ hdisk2            00-16-01-15,0     2.2 GB 16 Bit Differential SCSI Disk Drive
NETWORK
+ tok0              00-05             Token-Ring High-Performance Adapter (8fc8)
+ tok1              00-07             Token-Ring High-Performance Adapter (8fc8)
MISCELLANEOUS
+ rmt0              00-08-00-6,0      5.0 GB 8mm Tape Drive
```

Yet this information should be available to your customer in case they want it. The full hardware report should be included on a diskette or tape that will be handed to the customer along with the report. You should have obtained that information at the end of Chapter 7, "The Preparation Phase" on page 65.

In case your benchmark used a great number of machines, it will probably make your report clearer if you divide them into different groups, giving a common description for each group and pointing out how each machine in each group might stand out.

Try and be as precise as possible when identifying the hardware used to minimize misunderstandings.

9.3 Software Configuration

At the end of Chapter 7, "The Preparation Phase" on page 65, you should also have obtained a list of all the software that was installed on each machine. Use this information as a starting point and reformat it to indicate the major software components. Always ask yourself if the information you include in your report is useful. Do not include all the details in the report, but make sure they will be available to the customer in case they need it. This probably means copying them to a diskette or a tape.

Reorganize your information to emphasize what the different machines have in common and how each one may differ. Emphasize meaningful differences. If your benchmark includes backup and restore tests using such backup tools as ADSM, then the server should be easily identified from the clients.

Be sure to include the specific levels, versions and releases of all the installed software. You might have had to update some software during the benchmark or install temporary fixes. Be sure to mention those so that they are easily identified.

9.4 System Customization

When reading this part, the customer should be made aware of the work that has been done to customize the environment to their specific needs. Make sure you include the following:

- System parameters
- Application parameters
- Network parameters
- User creation

- Logical volume definitions
- File placement

9.5 Workload

How was the workload defined and implemented? The report should describe the workload. Typically, this should include the type of workload that was used, the transaction rates, the number of users and the think time. If your benchmark is Internet oriented, you should include the number of Web hits per hour. This list is definitely not comprehensive. Keep in mind that the idea behind this workload description is to provide the customer with all the information needed to understand the results, but not overwhelming them with data.

The workload description should not include all the script that have been created. Those should be included on the diskette or tape that will accompany the report.

9.6 Monitoring Tools

This part should answer the following questions:

- Which tools were used?
- Why were they chosen?
- How were the tools used?
- How should the output be read?

Make sure you include a statement indicating that the overhead generated from using these tools has been verified to be negligible.

9.7 Results

All the data that you have collected should be handed to the customer, but it should certainly not go into the printed report that they receive. Some selection must be made.

9.7.1 Keeping the Customer's Goal in Mind

Lets suppose:

- Machine N does 40 transactions per minute and costs $100,000
- Machine M does 30 transactions per minute and costs $50,000 system.

Which machine is the best? The answer could be *both*:

- Machine N can do more transactions per second.
- Machine M does more transactions per dollar.

This happens quite often in competitive benchmarks when the benchmark goals are unclear and the vendors bid different size machines. Hopefully, the definition and planning phases will have enabled you to clearly define what the benchmark goals are.

So which vendor wins the benchmark? That depends on the customer:

- If the customer wants the highest performance level then machine N wins.
- If the customer wants good price performance then machine M wins.

What this means is that knowing your customer and what they want is more important, in this case, than a good benchmark result. The customer may be after the best performance at any price or they may want the best performance for a given price. Refer to the benchmark definition phase and make sure that the results you have emphasized actually prove that you can fulfill the goal the customer has defined.

9.7.2 Emphasizing Results

Let's look back to the example mentioned in 2.7, "Presenting Results" on page 21. The point still remains that we can either present machines X and Y as ″Machine X is 50 percent faster than machine Y″ or ″Machine Y is 33 percent slower than machine X″. The impact may be sightly different depending on which way is chosen. We may *feel* that the first statement emphasizes how fast machine X is, whereas the second one minimizes how much slower Y is. Both statements being true, choose whichever one will put you in the best possible position.

9.7.3 Graphs, Charts and Tables

Make sure that you use the most appropriate way of presenting your results. As an example, graphs are usually the best way of presenting different activities, such as resource utilization, in time. Use colors that can be easily distinguished from one another to represent each of the resources being plotted. Be aware that sometimes color cannot be used and that there must still be a clear way of distinguishing one resource from another.

If what you are trying to represent is the way a particular thing can be divided up, then pie charts are very useful. This could be the case if you want to represent the relative importance, in percentage, of the different tasks that have to be executed during the benchmark.

If, on the other hand, you want to compare the elapsed time it took to complete the benchmark on five different configurations, you will be better off using a bar chart.

All graphics should be clearly labelled. For what is each color used? What does each axis represent and which unit does it use? Graphs should be kept simple. Do not try and plot your 500 measurements on one graph. What is important is that the general tendency should be respected.

Keep tables small. Large tables are very difficult to read and therefore loose impact. Do not try and reduce the font you use to put more into your table!

All your graphics should be followed by comments highlighting their purpose.

9.7.4 Comments and Recommendations

Did the benchmark go as planned? What are the reasons the execution may have gone askew? How was this corrected? What can be done to prevent future problems?

You might wonder if you should mention, for example, that a number of disks crashed during a benchmark. The answer might depend on whether you need to and whether you can show the incident in a positive way. One of our customer particularly appreciated the fact that we mentioned in the report that the crashes were due to an older version of the disk microcode being installed. Realizing this was very time consuming during the benchmark, but it saved the customer a lot of time when they went into production.

Have bottlenecks been identified that could not be overcome? Would removing one of the customer-imposed constraints have helped? For example, the tests you've undertaken may have shown that results could not be improved because you were not allowed to increase the amount of memory used. You may also inform the customer that a faster machine could have been used. In both cases you should assess what performance gain could be expected. However, be careful, since extrapolations should only be seen as hypotheses that have not been tested.

Has anything been noticed during the benchmark that could help the customer improve their environment? This, for example, could include any comment you may have on the database design or the way the queries are run.

9.8 Archiving the Results

Once your report has been written, you should remember to keep multiple copies of it. You should, at all times, be able to produce a new copy of your report. Remember that the others may lose their copy.

9.9 Presenting the Results to the Customer

The one thing you can be sure of is that, though writing a report is always a mandatory requirement, most customers will not find the time to read it. If you are lucky, they will browse the “Management Overview”.

Furthermore, there is a risk that, after weeks of careful planning and preparation followed by late hours running a benchmark and finally writing the results, the formal report is simply sent in the mail and the benchmark rapidly forgotten.

Often, during the benchmark, the customer will have taken notes, recorded (in a rough form) some of the results and formed their own impressions of the success level of the benchmark. But it has been found (by experience) that the customer and the benchmark center can have different opinions on the benchmark result, what they mean and the overall benchmark rating against the success criteria. A presentation will make sure the benchmark results are understood and the important points covered with the busy customer manager. It will ensure that both parties have the same understanding of the benchmark results and its implications. It also allows any concerns, issues or misunderstandings to be addressed and discussed.

Do not forget that this is also a good opportunity to sell the merits of the computer system involved.

9.9.1 It’s All in the Telling

As an example, the benchmark achieved 98 percent of the success criteria. This benchmark can be viewed as:

- A complete failure due to not achieving the success criteria.
- A complete success with information, clearly-defined problem areas and clearly-defined actions on how to achieve the last two percent or more.

Which of these impressions is correct, and which is related back to the customer’s management by the customer’s technical staff? This cannot be answered with any certainty, because this is a matter of personal judgement and impression.

9.9.2 The Highlights

The whole point of presenting the results to the customer is to allow the benchmark center team to place the results in context and to put the benchmark results in the most favorable light. Management makes key decisions based on small variations in numbers. However, real life is often more complex than this, and it needs to be talked through. For example:

- Are there actions to be addressed by the customer?
- Are there application improvements that can drastically improve the performance of the system?
- Does the database design need to be re-thought?
- Are there options in the configuration that need to be balanced against price and later upgrades when the workload grows beyond the initial installation?
- Are there systems of other sizes that were benchmarked, and trends that can help to size systems for different needs?
- Does further benchmark testing need to be performed?

9.9.3 Forced Feeding

The presentation needs to be carefully prepared.

- Make sure the customer's project sponsor is present.
- Make sure there are plenty of extra copies of the report.
- Make sure there is an agenda.
- Prepare foils of the:
  - Highlights.
  - Conclusions.
  - Actions.
- Make sure the presentation leads to discussion about the buying decision.

The end of the presentation for a successful benchmark is a major sales opportunity. This means it is very valuable to get the sales person involved in planning and deciding the points that should be covered so these dovetail nicely with the benefits the sales person knows to be important to the customer.

9.9.4 Alternative Presentation Method

Time and travel constraints often mean that the results cannot be presented in a face-to-face meeting. In this case, the presentation can be handled as a well-prepared, formal telephone conference call. Make sure that there is a precise agenda and that it goes through the major points to be covered.

Each participant should have a printed copy of the report at hand, so that graphs and tables can be easily referred to.

Chapter 10. The Follow-Up Phase

This phase starts after the benchmark report has been written and is a collection of tasks that the benchmark center should perform. With every mature organizational process, there must be a way to ensure the service is effective and feedback to make improvements to the service and the process. This is the main point of this phase.

After the benchmark report is written the following steps need to be performed:

- Sales followed up

  Typically, a potential sale initiates the benchmark process. Once the customer has the benchmark report, your sales person has an excellent sales opportunity.

- Service and process feedback

  Asking the customer for feedback is very constructive and can yield valuable information. This may have been performed before the benchmark report was written, while the customer was at the benchmark center or afterwards. Collecting and acting on feedback can help in the following ways:

  – To determine if the service (in this case a benchmark) was successful.

  – To determine if the customer bought the computer system or upgrade, since this justifies the benchmark center's work.

  – To establish if the process (in this case the benchmarking process) itself needs to be:

    - Changed or improved
    - Better structured and organized
    - Better communicated and/or documented

- Sell the benchmark center by writing a benchmark war story

- Complete and file all the benchmark information

- Return hardware or software on loan

This chapter covers these activities that have been found to work in practice and to increase customer satisfaction.

10.1 Sales Follow-Up

From experience, sales people attached to the customer who are using the benchmark center are either very pro-active and asking for updates every hour or are busy elsewhere are not involved in the benchmark at all.

If its a pro-active sales person they are likely to lead the benchmark presentation and ask you to be prepared to cover the technical aspects.

If the sales person is not involved, then its the benchmark center's role to get sales involved to close the business.

10.2 Customer Feedback for Improvements in the Service and Process

Feedback forms have become a way of life in many large organizations, but their value should not be underestimated. They are an effective way to ensure communication with the benchmark customer and gain a useful insight into the way the benchmark is viewed by its customers.

10.2.1 Feedback Form Contents

These feedback forms should be very carefully written. Often benchmark centers are measured on the yearly customer satisfaction rating and the only means to measure this is through a feedback form from which a number can be extracted. Therefore, write a feedback form that allows the customer to rate:

- The benchmark
- The benchmark service
- The benchmark center

A scale is also needed. Make sure that the scoring scheme is fair (that is, the score actually ranges from 0 to 100 percent satisfied) and that customers are encouraged to be honest in their evaluation. Make sure there is space for more general comments. These comment are the most useful information to make improvements in benchmarks in the future. Also include space for rating and commenting on other vendor's benchmark centers; this can yield useful information about competitors.

See Appendix B, "Checklists and Sample Forms" on page 113 for an sample benchmark feedback form.

10.2.2 Feedback Form Information

Benchmark feedback forms can provide many types of information:

- They establish if the customer is satisfied.

If the customer is not satisfied, this can be addressed immediately to conclude the benchmark on good terms.

- Any good ideas that customer suggested can be addressed.

  This often highlights anything that should normally have happened but, by mistake, was forgotten. This often includes excellent new ideas that would improve the center and its service.

- Long term evaluation.

  The forms from many benchmarks can be saved and re-evaluated over a longer time period. We suggest quarterly or at a minimum once a year. This allows longer trends to be spotted and repeated items to be addressed. If many customer complain about what you might consider an insignificant point, then it should be sorted out.

10.2.3 Feedback from Whom

The people attending the benchmarks (typically the customers and technical staff) are not usually the people paying for the service, nor are they the end users of the computer system once it is put in production. They will perceive and rate the benchmark center in detailed practical terms such as:

- Is the tea, coffee and lunch free?
- Is the office space good and is the access to telephones, FAX and dial-out connections for laptops adequate?
- How easy is making travel arrangements and booking local hotels?
- How long did it take for benchmark center staff to address technical problems?

This means that you may want to consider getting feedback from both the technical people and from the management-level people who are paying for the benchmark and who will make the buying decisions based on the benchmark results.

The ultimate feedback is, of course, a sale.

10.3 Benchmark Success War Story

One important (though often unrealized) task of a benchmark center is that it must sell itself, so that other parts of the organization understand its value and learn from its successes. Perhaps the best proof of performance is when the customer buys a system as the result of a benchmark.

You will be amazed at the interest these generate and the value customers place on learning from the experience of others, even their own competition, and hearing a good story.

The success story does not have to be large but must be very readable and non-technical. Typically, half a page in length. It should include:

- Customer name.
  - This must be agreed with the customer in advance or use a phrase like “a well known name in the so-and-so business.″
  - The customer’s name can still be used on internal documents.
- Type of business they are in.
- Time and value of the sale.
  - A simple year and month plus rough estimate of the system cost including service, training, facility management, and so on.
- The type of system involved.
  - For example, OLTP, DSS, network computing plus accounting, telesales, customer support, human resources and so on.
- Brief details of the benchmark configuration
  - Just the system type, memory, disk size and tape drives.
- Brief details of the results.
  - Number of users or the batch throughput.
- Compare the above to the success criteria.
  - A simple statement, ″This was XX percent above that required.″

If the benchmark was head-to-head against the competition, then also include:

- How much the competition was behind, if known.
  - Keep this simple - ″Our machine managed NN more users or the batch run was completed in MM less time.″
- Why your machine/benchmark was better.
  - For example, ″Our machine has a more powerful CPU, the disk subsystem gives us a NN percent advantage, particular applications are well-tuned or suited for our platform.″
  - Also highlight any known weaknesses in the competition. For example, ″Any reliability issues (did their system crash) or connectivity issues.″

  - Or, ″Our benchmark team technical skills are superior.″
- If particular techniques were used and found to make a large difference in performance, do not keep this a secret.
  - Only include these techniques if the audience is technical.
  - Briefly describe the technique and its effect, but state where more information can be found on a more permanent place (usually a Web site).

The success story should be widely published internally and externally by all available means, thus ensuring the benchmark center's status and value as part of the sales and marketing effort.

A failure story can be useful to so that others do not have the same problems, but note that failed benchmarks are almost always due to poor planning and preparation and not hardware or software problems.

10.4 File the Documentation

Once the benchmark is finished, make sure that you complete all the documentation and file all the benchmark information in a place where it can be retrieved.

Benchmarks are often repeated later or further detailed technical questions were raised, sometimes within weeks or years later. The next benchmark usually has something slightly different, perhaps a new release of the application or an alternative database or a different hardware configuration. Saving all the information on this benchmark will save many hours and get technical people up to speed quickly on what the issues are likely to be. Customers also like to think they are known to the center even if some of the faces have changed in the interim.

Also, there are legal requirements to satisfy. The benchmark results might be contested when there is a problem with the system going into production or the benchmark was not paid for. Only by keeping full records and having a copy of everything, can these problems be sorted out.

When all the paper work is available it can be established that the production system hardware, software, workload, database data and configuration are widely different and this is responsible for the difference in performance.

The following items are a minimum:

- List of contact telephone numbers, FAX, e-mail and so on.

- The signed contract
- The details of the invoice and payment
- The benchmark request form
- The benchmark risk review
- Every FAX
- Every e-mail
- Notes from all telephone calls
- The benchmark report
- Anything exchanged between the customer and the benchmark center
- All checklists and action lists
- The feedback form
- Results, configuration and scripts on tape
- Any backup on tape
- Spreadsheets used to generate graphs for reports

Various office systems can assist in keeping all these details and information together in some sort of project file.

10.5 Return Hardware or Software on Loan

Any hardware or software on loan to the benchmark center must be returned.

It is worth formally tracking the hardware and software when it arrives, to ensure that the same equipment is returned and the owner acknowledges it was returned in full.

Appendix A. IBM RS/6000 Benchmark Centers

To assist IBM RS/6000 sales representatives and IBM business partners in realizing benchmark projects as discussed throughout this book involving IBM RS/6000 servers and workstations, IBM offers support through some RS/6000 benchmark centers in various regions of the world.

A.1 Non-Standard Custom-Defined Benchmarks

In EMEA, these centers are usually part of the local IBM RS/6000 Technical Support organization from which they receive the resources they offer. These include

- Hardware (subject to local availability)
- Software (usually basic AIX installation and monitoring tools)
- Support (according to local guidelines)

As an IBM RS/6000 sales representative or IBM business partner, we recommend you visit the Web site of your local IBM RS/6000 technical support organization and find out about the offerings they provide and the people to contact to get more detailed information about the process to follow.

Here are some of the Web sites that were available during the time this book was produced:

- RS/6000 Benchmark Center - Bedfont Lakes, UK

  `http://w3.aixncc.uk.ibm.com`

- RS/6000 Benchmark Center - Montpellier, France

  `http://w3.pssc.mop.ibm.com`

- RS/6000 Benchmark Center - Munich, Germany

  `http://www.munich.ibm.com/ts.htm`

- RS/6000 SP Benchmark Center - Poughkeepsie, New York, USA

  `http://w3.vendor.pok.ibm.com`

- RS/6000 Benchmark Center - Westlake, Texas, USA

  `http://bench.aix.dfw.ibm.com`

Note: Addresses starting with `http://w3...` are on the IBM intranet and are not accessible from outside IBM.

A.2 AIX Performance Group Benchmarking Activity

The AIX Performance Group uses a suite of benchmarks to analyze AIX and RS/6000 system performance. TPC-C, TPC-D, SPECfp, SPECint, SPECweb, LADDIS and LotusNotes are included, among others. These benchmarks are used in a number of ways:

- AIX regression testing

  When an AIX release is under development, new levels of the code are tested throughout the development cycle. A series of benchmarks, performed on a consistent set of hardware, is used to test the different AIX levels. This regression benchmarking detects possible performance degradations incrementally as new features are added to AIX.

- Performance assessment of new RS/6000 systems

  The AIX Performance Group plays an integral role in the development of new RS/6000 systems. Typically, a performance goal is set for new systems based on a projected benchmark result. As soon as the system is capable of supporting a workload, benchmarks are run to assess the performance. This performance assessment continues through the end of product development.

- Marketing tool

  Many of the benchmarks used have high visibility in the industry. If there is a positive marketing story (for example, performance or price/performance leadership), the benchmark results are exploited.

The level of benchmarking done by the AIX Performance Group requires a significant investment in resources. Dedicated benchmark teams are staffed for each benchmark. A multi-million dollar lab is maintained to support benchmarking activities. Partnerships with third-party vendors are established to help drive improvements in non-IBM products that are integral parts of the benchmarks. IBM performance engineers are occasionally placed at database vendor sites to help with these improvements. Active participation in benchmark consortiums is essential to make sure that IBM interests are considered.

Most of the benchmarks used by the AIX Performance Group are industry standard benchmarks. Unlike custom benchmarks, industry standard benchmarks are developed by a standards group that oversees the execution of the benchmark and grants approval of benchmark results. Every published result is subject to a review process before it is deemed valid. For example, TPC results require the approval of a TPC-certified auditor. An auditor′s attestation letter must accompany the results when they are submitted for publication. This strict policing of the benchmark helps ensure the integrity of the benchmark results and a consistent implementation across different hardware platforms. For these reasons, industry standard benchmarks are the best way to compare the performance of different systems.

Benchmarks are valuable tools for measuring and understanding system performance. However, if used improperly, benchmark results can be misleading.

Standard benchmarks (that is, TPC and SPEC benchmarks) are best used to compare the performance of one system against another. They in no way guarantee that a particular performance level can be met in any environment other than the exact one used for the benchmark. To use benchmark results to predict system performance, you must know the specifics of the benchmark and compare them to the workload that will actually run on the system. Moreover, you must quantify these differences and know how they will affect performance. As you might imagine, this is not an easy task.

Benchmark configurations are extremely important to the performance of a system. Disclosure reports for the industry standard benchmarks must include detailed information about the hardware and software configuration. Subtle changes to the configuration could have a dramatic affect on performance.

For example, in many benchmarks maximum throughput is achieved by driving the CPU as close as possible to 100 percent, so the system is configured with enough hardware to eliminate bottlenecks. This includes memory, disks and front-end client systems for TPC-C. Therefore, some of the benchmark configurations include much more hardware than what the typical customer would configure.

The AIX Performance Group and other IBM benchmarking groups have invested much in running and publishing benchmark results. Take advantage of the work that has been done but understand the results and use them with caution.

Appendix B. Checklists and Sample Forms

This appendix is intended to help you prepare for a successful benchmark. In this appendix you will find:

Checklists	The checklists are useful reminders of the important issues and points.
Agendas	The agendas are useful reminders during a meeting, detailing the issues that must be covered.
Sample forms	The sample forms are useful as they contain all the questions you will need to answer and show the level of detail required.
Example forms	These are the same as above but with example answers.
Legal letters	Also included are two sample letters covering the important legal issues involving copyright release so that non-IBM software can be used in the IBM benchmark center.

You should contact your local benchmark center for the forms to use. Note that many are actually online Web-based forms.

B.1 Sample Benchmark Nomination Form

This is a sample benchmark nomination that is filled-in by the benchmark requester and returned to the benchmark center. Submission of the form does not commit either party to actually run the benchmark, but it formally starts the benchmark dialogue and process.

Do not use this form, but check with your local benchmark center and request their actual form.

B.1.1 Part 1 - General Information

General information should include the following:

- Customer name.
- Location (street, City, State, Country).
- IBM organization details: (division number, department number, area number).
- IBM OMSYS opportunity number.
- What is your assessment of the chances of winning (in a percentage).
- Who are the competitors for this opportunity?
- What is the revenue potential of this opportunity?

  - This year? $K Dollars.
  - Next year? $K Dollars.
  - Succeeding years? $K Dollars.
- Who is funding this benchmark (IBM/Customer/Business Partner)?
- Time frame when benchmark facilities are requested.
  - Start date (mm/dd/yy).
  - End date (mm/dd/yy).
- What are the objectives of this benchmark?
- What is to be measured in this benchmark and how will success be determined?
- What will IBM gain by doing this benchmark?
- Identify the general hardware, software, skills and workload requirements.
- Contacts - name, telephone number and e-mail address.
  - Management contact.
  - Marketing contact.
  - Technical contact.

B.1.2 Part 2 - Detailed Information

Some benchmark centers include the following extra questions as part of the benchmark nomination form, while others cover these questions through a hardware requirements form nearer to the benchmark planning meeting. Either way, these details will have to be addressed before the benchmark starts.

Please complete just one of the SP or Uniprocessor/SMP sections below:

- SP benchmark machine.
  - Node type.
  - RAM per node.
  - Internal disk per node.
  - External disk and disk type per node.
- Uniprocessor or SMP benchmark machine.
  - Machine type.
  - Number of CPUs (SMP).
  - RAM.
  - Internal disk and disk type.
  - External disk and disk type.
  - Network type.
- Are there any additional hardware requirements for the benchmark (such as special adapters, attached tapes, or extra workstations) and why are they needed?
- Are there any special software requirements for this benchmark?

Include the following questions for technical or scientific benchmarks only.

- What applications will be run in this benchmark?

The following questions are for commercial or database benchmarks only.

- RDBMS name and version.
- How much raw user data (excluding indices) is to be used in this benchmark? (Gbytes)
- Has a consultant been hired to monitor the benchmark? If so, who is it?

Workload and data.

- How many concurrent active users are there for:
  - Online Transaction Processing (OLTP)?
  - Simple DSS?
  - Complex DSS?
  - Batch/report?
- Will emulation of user sessions be required and the terminal type?
- Where is the benchmark database data coming from?
  - If it will be generated:
    - From what application?
    - Ramping up sample data?
  - If from an existing database:
    - Database and machine?
    - What media format will be used?
- Has the application been ported and tested on AIX and with which version?
- Are systems management and RAS demonstrations part of the benchmark, and if so, what are the details?

B.2 Sample Benchmark Planning Meeting Agenda

Use this agenda for the benchmark planning meeting to make sure all the important and vital points are covered. Either the facts are documented during the meeting or attendees accept a task to find the answer before the benchmark starts.

1. Account background/status and customer's business
2. Customer objectives (in customer's words)
3. Success criteria
   - Measurement metric
   - Preset success bars
   - Measurement tools
4. Benchmark workload definition
   - Intended real use of the system (application description)
   - Benchmark workload - batch/interactive or both
   - Variances

- Workload/test duration
- Data reset between tests

5. Benchmark requirements

   a. Hardware requirements CPUs, memory, disk, tape, network, other

      - System under test (SUT)
      - Remote Terminal Emulator system (RTE)
      - Connected systems

   b. Software requirements including version levels

      - Operating System (OS)
      - IBM Licensed Program Products (LPP)
      - Relational Database Management System (RDBMS)
      - Third-party copyrighted code
      - Customers own code

   c. Skills requirement

      - Project management
      - Benchmarking
      - Systems administration
      - Operating System tuning
      - RDBMS DBA
      - RDBMS tuning
      - Application administrator
      - Application owner
      - User simulation expert

   d. Data requirements

      - Size and complexity
      - Data source and conversions
      - Data generation
      - Media and tape format

6. Benchmark activities workplan
   - Half-day incremental plan of activities
   - Responsibilities
   - Decision points
   - Contingencies, back out, alternatives
7. Action list
   - Action items
   - Who is responsible
   - Response due dates

B.3 Sample Benchmark Project Workplan

A benchmark plan includes three sections:

1. Definition
2. Preparation activities workplan
3. Benchmark activities workplan

A brief outline for these benchmark plan elements and an example benchmark activities workplan are presented here. These are intended to be
used as general guidelines and need to be tailored for individual benchmark requirements.

B.3.1 Definition

Benchmark objectives, workload, and success criteria must be defined. The environment details and configuration for hardware and software must be documented to provide a base of understanding for all benchmark team participants.

B.3.2 Preparation Activities Workplan

The following items should be included in the preparation activities workplan.

- Identify benchmark team.
- Identify and acquire needed skills.
- Acquire unique hardware.
- Negotiate third-party copyright agreements.
- Identify, acquire, and test customer application programs and data.
- Convert and test non-IBM customer application programs.
- Identify, design, develop, and test proposed applications.
- Prepare and test utility procedures to define, load, and delete databases.
- Develop and test ease-of-use procedures and scripts.
- Ensure knowledge of performance tools usage and reporting.
- Ensure knowledge of hardware and system operation.
- Determine method and timetable for porting/testing applications.
- Plan for customer or third-party data and program security.
- Identify customizing requirements (user IDs, file placement, authorization, and so forth.
- Communicate software customizing requirements to benchmark support.
- Prepare to use Remote Terminal Emulation (RTE).
- Plan scenarios that can be re-executed with minimal intervening set up or clean up.
- Create interactive application scenarios for RTE script input.

- Define parameters to generate RTE response time reports.
- If available, use appropriate tools to code the RTE scripts.

B.3.3 Sample Checklist for Benchmark Activity Workplan

Use the following as a starting point for the list of required activities. Ensure that the environment provided is correct; that it contains all required hardware, software, files (correctly placed), network connections, print capabilities, phone lines, and other facilities needed.

- Ensure that all required user IDs are defined, working properly and have correct authority levels, file permissions, and so forth.
- Ensure that any system defaults that are inadequate are specified, including maxuproc, maxdata, maxfs, no parameters and TCP/IP parameters.
- Install any required non-IBM code.
- Perform any required customization for applications, such as Oracle, Sybase, and so forth.
- Install customer applications and data.
- Build customer databases.
- Test applications (using real terminals).
- Create and debug any required RTE scripts. Always use the live terminal
  to test a task prior to attempting to script it, so that you know it works prior
  to script capture.
- Run RTE with one user to validate scripts.
- Test report generation for RTE, vmstat, iostat and PTX/6000 as required.
- Run iterative tests collecting timing data and tuning bottlenecks as required on operating system, applications, databases, I/O files, and network. Consideration should be given to the following items:
  - File placement, location of frequently used database tables, and so forth.
  - Paging space size and location. Monitor amount of paging that occurs.
  - Compiler options and any application modules that consume large amounts of time or resources. Do hot spot analysis as required.
  - I/O buffering and amount of waiting on I/O, the disks that are waiting on I/O, determine which files reside on any offenders.
  - I/O blocking factors, use of JFS versus raw mode file systems.
  - Any applicable application tuning parameters.
  - Any applicable network tuning parameters.
- Determine the number of timing runs the final benchmark will require.
- Determine amount of time that will be required for each timing run, including test preparation, execution, data collection, and restoring the environment/data for the next run.

- Determine how many hours per day will be worked and estimate as closely
  as possible the number of timing runs that will be made each day.
- Verify that the timing runs are repeatable before starting iterative tests.
- Prepare on-site customer presentation of testing results, if required.
- Ensure all mandatory tests are run, including any functional demonstrations, if required.

B.3.4 Example Benchmark Activities Workplan

This example begins on a Monday and includes a half-day of hardware setup. It only documents activities through the following Thursday. However, this illustrates the purpose and some of the content that a plan should have.

All time frames are approximate.

- Day 1 (Monday AM).
  - Mid-day arrival of benchmark team.
- Day 1 (Monday PM).
  - Orientation.
  - Review benchmark activities workplan.
  - Start loading binaries.
  - Start loading data.
- Day 2 (Tuesday AM).
  - Continue loading/restoring data.
  - Install and configure program binaries.
  - Restore any utility libraries.
- Day 2 (Tuesday PM).
  - Back up data for subsequent use.
  - Start unit test of application(s) using real terminal.
- Day 3 (Wednesday AM).
  - Begin using RTE to capture/code scripts.
  - Begin customizing/tuning captured RTE scripts.
- Day 3 (Wednesday PM).
  - Continue capturing and editing/testing scripts.
- Day 4 (Thursday AM).
  - Complete coding/testing of individual RTE scripts.
- Day 4 (Thursday PM).
  - Run each RTE script against system under test (SUT) for single terminal emulation.
- Day 5 (Friday AM).
  - Complete RTE script testing for single terminal emulation.
  - Expand RTE to multiple terminal emulation.
- Day 5 (Friday PM).

  - Run a mix of RTE scripts against SUT for multiple terminal emulation.
  - Debug script interaction, conflicts, record locking, and timing problems.
  - Make initial timing runs.
- Day 6 (Monday AM).
  - Review initial timing runs.
- Day 6 (Monday PM).
  - Implement tuning efforts for next RTE timing run (move files, increase buffers, modify RTE scripts as needed).
- Day 7 (Tuesday AM).
  - Rerun RTE scripts for timing run.
  - Start ramping up number of users and collecting performance data.
- Day 7 (Tuesday AM).
  - Retune and rerun RTE scripts adding more users.
- Day 8 (Wednesday AM).
  - Complete RTE runs for the first SUT.
  - Save all output performance data.
- Day 8 (Wednesday PM).
  - Back up and restore system on second SUT.
  - Continue RTE runs on second SUT.
- Day 9 (Thursday AM).
  - Back up and restore system on third SUT.
- Day 9 (Thursday PM).
  - Continue RTE runs on third SUT.
  - Complete activities at benchmark center.

As can be seen, this is a relatively simple half-day plan that only deals with tasks, not the individuals or teams who have responsibility for performing the tasks. The plan could be expanded to appear more like a grant chart, which describes various activities occurring in parallel, performed by different teams or individuals, with the parallel efforts then converging at some point. This would maximize efficiency in a very complex project. For instance, instead of taking three weeks to complete all activities sequentially, the overlap of non-interdependent tasks might permit completion in two weeks, with the effect of saving a large percentage of cost (for example, equipment usage and hotel and living expenses for the attendees).

However, this complex parallel plan must be thought out in great detail to prevent having a breakdown in continuity which would negatively impact the project.

It should always be remembered that the project is only as good as the planning permits.

B.4 Benchmark Participants' Responsibilities Checklist

This documents the responsibilities of all participants in a benchmark. It has been our experience that the division of responsibilities as it is described in our example, and which has evolved over many years, is one which yields a great degree of success in a benchmark project.

The key participants in the benchmark process are:

- The marketing team or customer account team
- The customer
- The benchmark support team

Each group is expected to assume their responsibilities as agreed upon and listed in this document, to ensure project success.

B.4.1 The Marketing Team and Customer Responsibilities

The customer is ultimately responsible for the success of their benchmark. The customer and/or IBM marketing team can use this checklist as a guideline for activities in their area of responsibility.

- Communicate detailed benchmark requirements to benchmark support.
- Clearly define the benchmark objectives, workload, and success criteria.
- Set realistic expectations regarding system performance.
- Staff the benchmark team with individuals having the appropriate skill level for the operating system and application-enabling software components.
- Coordinate the procurement and installation of required systems and I/O not available at benchmark site. This includes removal of hardware and software at the conclusion of the benchmark.
- Develop a detailed benchmark activities workplan and present it to benchmark support prior to the readiness review.
- Assign tasks and track progress of benchmark preparation activities.
- Coordinate customer and third-party participation.
- Prepare to execute the benchmark. Tasks may include:
  - Application program and data conversion.
  - Application development and testing.
  - Defining the benchmarking run scenarios.
  - Discussing considerations for program and data portability and security.
  - Completing non-IBM software copyright agreements.
  - Participate in a readiness review.
- Execute the benchmark. Tasks may include:
  - Installing non-IBM software.
  - Customizing system and application software.
  - Integrating the customer′s workload.

  - Systems operation.
  - Terminal emulator script generation.
  - Problem determination and resolution.
  - Application performance analysis and tuning.
- Provide benchmark results to benchmark support (if requested).

B.4.2 Benchmark Center Responsibilities

The following items should be included in the list of benchmark center responsibilities.

- Support the marketing team and customer during the definition of benchmark objectives, workload and success criteria.
- Review nomination and determine benchmark feasibility.
- Refer marketing team to alternate sources of support (if appropriate).
- Review and provide guidance on benchmark specifications.
- Provide an estimate of the cost of the benchmark.
- Provide guidelines for preparing and executing the benchmark.
- Conduct the benchmark planning session (if applicable).
- Review the benchmark plan and conduct the readiness review.
- Provide facilities.
- Provide supported IBM hardware and software.
- Coordinate installation of additional hardware for specific benchmark requirements.
- Set up and install supported IBM hardware and software prior to benchmark team arrival.
- Provide project management skills during benchmark preparation and execution.
- Provide the AIX operating system environment.
- Provide AIX operating system performance evaluation and tuning assistance.

  This assistance is provided on an as-available basis during normal benchmark support hours only. Any other assistance must be specifically contracted for.

B.5 Readiness Review Checklist

The readiness review checklist is typically used a week before the benchmark start date to confirm everything is in place and will be ready.

- List the quantifiable objectives of the benchmark.
- All data and program conversion complete.
- All applications have been tested, debugged and tuned.
- All non-IBM copyright releases are negotiated and signed.
- User simulations required:
  - All run scenarios are detailed.

    - All scripts are coded, if possible.
- Detailed benchmark plan agreed upon.
- Benchmark activities workplan agreed upon.
- Benchmark team is adequately staffed and prepared to assume responsibilities.
- List benchmark attendees:
    - Name.
    - Company.
    - Contact.
    - Title or benchmark role.
    - Dates, if not full duration.

B.6 Sample Copyright Release Letters

To installing any non-IBM copyrighted software at the benchmark center, a written copyright release must be received at the center before the benchmark start date. This is a mandatory requirement. The benchmark center will not violate copyright laws.

The following two release letters have been approved by IBM general counsel and intellectual assets (patents). If the appropriate release letter is not acceptable to the owner of the software copyright, it is a marketing team responsibility to negotiate a unique agreement between the copyright owner and the benchmark center.

Use the **Customer Copyright Release** when the customer owns the copyright to the software required for the benchmark.

Use the **Third-Party Copyright Release** when a third-party vendor owns the copyright to the software required for the benchmark. The customer should negotiate the software release and send this letter to the benchmark center.

These sample letters should be typed on the customer′s letterhead exactly as shown below. Any changes to the content (other than the names below) will require IBM legal approval.

Replace:

- <ABC Corporation> with the customer′s name
- <XYZ Inc> with the third-party's name (3pd part form only)
- <XXXX> with the software package name(s)
- <BMN> with the benchmark center's manager's name
- <BCA> with the benchmark center's address

B.6.1 Customer Copyright Release (Customer Owns the Copyright)

The following is the Customer Copyright Release sample letter.

<BMN>
RS/6000 Benchmark Center
<BCA>

Dear <BMN>,

<ABC Corporation> will be using its copyrighted software package
<XXXX>
during the upcoming benchmark at the above benchmark center.
<ABC Corporation> warrants that the installation of this package will not
constitute a violation of <ABC Corporation>′s copyright. The package will
be
installed and used for this benchmark only.

<ABC Corporation> will be responsible for:

- Installation
- Integration
- Problem determination
- Problem resolution
- Deletion from the IBM system of the <XXXX> package

IBM will be responsible for the installation, problem determination, problem

resolution, and tuning of all supported IBM software under which the
<XXXX>
package will execute.
IBM will be given no material by <ABC Corporation> that is the
confidential or

proprietary information of <ABC Corporation> or any other party. In no
event
will IBM be given any source program code.

Yours sincerely,
Authorized Representative of <ABC Corporation>

B.6.2 Customer Third-Party Copyright Release

The following is the Customer Third-Party Copyright Release sample letter.

<BMN>
Benchmark Center

<BCA>

Dear <BMN>,

This letter will certify that <ABC Corporation> has the permission of
<XYZ
Inc> to install <XYZ Inc>′s copyrighted software package <XXXX> at the
above benchmark center.
<ABC Corporation> warrants that the installation of this package will not
constitute a violation of <ABC Corporation>′s licensing agreement with
<XYZ
Inc.>
The package will be installed and used for this benchmark only.
Either <ABC Corporation> or a duly authorized <XYZ Inc> representative
will
be responsible for:

- Installation
- Integration
- Problem determination
- Problem resolution
- Deletion from the IBM system of the <XXXX> package

IBM will be responsible for the installation, problem determination, problem

resolution, and tuning of all supported IBM software under which the
<XXXX>
package will execute.
IBM will be given no material by <ABC Corporation> that is the
confidential or

proprietary information of <ABC Corporation>, <XYZ Inc>, or any other
party.

In no event will IBM be given any source program code.

Yours sincerely,
Authorized Representative of <ABC Corporation>

B.7 Alternative High-Level Benchmark Plan Checklist

This section contains an alternative checklist for project managers as used in the Poughkeepsie benchmark center. This may be of help as its is more prescriptive of the tasks that are involved. Note that this benchmark center deals with RS/6000 Scalable Parallel (SP) machines and includes items specific to the SP benchmarks.

Use this work sheet as a guide to planning your commercial benchmark project. Some of the items may not be applicable to your benchmark. We may also have missed some activities that are required by your benchmark.

Table 6. Benchmark Project Manager's Checklist

Activity	Expected Date	Completed Date
Benchmark goals defined		
System resources defined		
Benchmark request submitted		
Benchmark support team committed		
Database logical design completed		
Application design complete		
High-level plan completed		
Benchmark plan review/team kick off		
Hardware availability plan complete		
Database physical design complete		
Application tested		
Day-by-day plan complete		
Data loading migration strategy defined		
Detailed plan/readiness review		
Hardware committed		
Data tapes to benchmark center		
Tapes loaded to MVS host (if required)		
Prepare for visit to benchmark center		
Benchmark system ready		
Database code loaded on system		
Team arrives at Benchmark Center		
Start benchmark		
End benchmark		
Post mortem at end of benchmark		
Benchmark report completed		

B.8 Alternative Detailed Benchmark Plan

This is the detailed benchmark planning checklist that matches the above high-level project manager checklist. This breaks down each activity into a number of smaller tasks and assigns primary and secondary responsibility to each task.

Table 7. Detailed Benchmark Plan Checklist Key

KEY	Skill	From
A	Account/Customer	BM team
D	Database consultant	BM team
P	Project manager	BM team
C	Benchmark manager	BM center
M	MVS data load team	BM center
H	Hardware team	BM center
N	Network team	BM center
S	System administrator	BM center

The following checklists are for the benchmark project manager to track the progress of each phase of the benchmark.

Table 8. Detailed Benchmark Project Manager's Checklist Section 1

Primary Function	Secondary Function	Tasks
Benchmark Goals Defined		
A		Understand objectives and scope of effort.
A		Identify customer objectives.
A		Identify success criteria.
A		Obtain definition of work - queries, tables, and so on.
A		Determine time frame of proposed benchmark.

Table 9. Detailed Benchmark Project Manager's Checklist Section 2

Primary Function	Secondary Function	Tasks
System Resources Defined		
A		Obtain definition of system - Nodes/CPUs, disk, tapes, host, memory, network, and so on.
A		Determine all software needed and the required release levels.
A		Determine if license agreements are in place for all software.

Table 10. Detailed Benchmark Project Manager's Checklist Section 3

Primary Function	Secondary Function	Tasks
Benchmark Request Submitted		
A		Benchmark form filled-out and submitted.
CA		Benchmark center call to review request.
AC		Discuss cost of proposed benchmark.
A		Forward account codes or ICA number to benchmark center.
C		Benchmark manager assigned by the benchmark center.

Table 11. Detailed Benchmark Project Manager's Checklist Section 4

Primary Function	**Secondary Function**	**Tasks**
Benchmark Team Committed		
A		Understand the role of any third-party consultants.
A		Understand the role of third-party code other than database.
A		Project manager assigned to run benchmark according to schedule and make decisions.
AP		Build benchmark team that can support itself.
AP		Obtain essential personnel to cover all planned shifts.
AP		Database MPP organization support person available for benchmark.
AP		Assign a database person who will support the database.
AP		Assign an AIX person who will support the system.
AC	P	Set a date for initial high-level planning meeting.

Table 12. Detailed Benchmark Project Manager's Checklist Section 5

Primary Function	**Secondary Function**	**Tasks**
Database Logical Design Completed		
PD	AC	Detailed review of customer's benchmark documentation.
AD	P	Determine need for TP monitor such as Tuxedo, Encina, CICS and update plan.
AD	P	Database planning and definition
AD	P	Definition of each table; record length, rows, indexes, and so on.
AD	P	DDL for each table obtained.
AD	P	Estimate space for table, indexes, rollback, temp, system
AD	P	Define space needed for staging area (equal to raw data).
AD	P	Define physical layout of the datasets and the plan for striping the data.

Table 13. Detailed Benchmark Project Manager's Checklist Section 6

Primary Function	Secondary Function	Tasks
Benchmark Plan Review		
P		Forward high-level plan and benchmark documentation to benchmark center.
C		Benchmark center team assigned - sys admin, data management, hardware support, database support, network support.
PC		Schedule benchmark plan review (kickoff meeting).
PC	A	Introduce benchmark team members.
AP		Review business opportunity behind benchmark request.
PC	A	Document the outstanding problems and issues.
CD	P	Determine if remote access to benchmark center will be needed.
PC	A	Determine the risk involved.
C		Determine benchmark center availability.
PC	A	Establish date for next review and work items to be done.

Table 14. Detailed Benchmark Project Manager's Checklist Section 7

Primary Function	Secondary Function	Tasks
Hardware Availability Plan		
HC	P	Identify and document all of the hardware needed to execute the benchmark.
HC	P	Develop plan for obtaining the required hardware and availability schedule.
PC		Obtain schedules for delivery of loaned equipment.

Table 15. Detailed Benchmark Project Manager's Checklist Section 8

Primary Function	Secondary Function	Tasks
Database Physical Design Complete		
DA	P	Determine logical volume names.
DA	P	Determine VSD parameters (if required).
DA	P	Build AIX scripts to create LVMs and create the VSD configuration file (if required).
DA	P	Build SQL scripts to create database and define tablespaces (including system and temp storage).
DA	P	Create/obtain the DDL to be used to define the tables.
DA	P	Build SQL scripts to run the queries.

Table 16. Detailed Benchmark Project Manager's Checklist Section 9

Primary Function	Secondary Function	Tasks
Application Design Complete		
A	P	Final or sample queries obtained.
A	P	Sample data obtained.
DP	A	Determine strategy for prototyping with RS/6000, cluster, and so on.
P	D	Locate resources for prototype effort.
P	D	Determine need for non-IBM licensed software.
D	P	Database planning and definition of prototype test.
D	P	Set up prototype system.
D		Create database table space.
D		Load data and create indexes.
D		Run the queries to validate that they are correct.

Table 17. Detailed Benchmark Project Manager's Checklist Section 10

Primary Function	**Secondary Function**	**Tasks**
Day-by-Day Plan Complete		
P		Create spreadsheets to assist building tables and indexes, running queries, and recording results.
P		Develop detailed plan and schedule.
P	A	Review of plan by initial team participants and update as agreed.

Table 18. Detailed Benchmark Project Manager's Checklist Section 11

Primary Function	**Secondary Function**	**Tasks**
Data Loading/Migration Strategy Defined		
AM		Establish overall data unload, load, and backup/restore plan for the benchmark.
PM		Review best means of unloading data with benchmark center.
AM		Review backup/restore plan with benchmark center.

Table 19. Detailed Benchmark Project Manager's Checklist Section 12

Primary Function	**Secondary Function**	**Tasks**
Detailed Plan and Readiness Review		
PC	A	Review plans for completeness and resolution of outstanding problems and issues.
PC	A	Review team staffing and skills
PC	A	Review status of hardware and software schedules.
PC	A	Assess risks involved (if any).
CP	A	Go or No Go decision made on whether to continue with effort.
C		Hardware committed by benchmark center.

Table 20. Detailed Benchmark Project Manager's Checklist Section 13

Primary Function	**Secondary Function**	**Tasks**
Data Tapes to Benchmark Center		
A	H	Build JCL or scripts to download data.
AH		Unload data and save JCL and system output.
AH		Ship tapes and system output to benchmark center.
AH		Notify benchmark center of shipment, air bill number and carrier.
H		Data loaded onto MVS system (if required).

Table 21. Detailed Benchmark Project Manager's Checklist Section 14

Primary Function	Secondary Function	Tasks
Prepare for Visit to Benchmark Center		
P	C	Send welcome document for travel, hotel information, and directions.
P	C	Send complete list of benchmark team members to benchmark center.
P	C	Notify benchmark center when customer is expected to arrive.
P	C	Set up briefing (if required during customer visit).
ND	C	Test remote access to benchmark center.

Table 22. Detailed Benchmark Project Manager's Checklist Section 15

Primary Function	Secondary Function	Tasks
Benchmark System Ready		
H		Install disk in racks (if required).
HN		Set up required hardware configuration and attach disk.
S		Install AIX and build system.
SM		Test system and MVS connection.
MS		Exercise disk.

Table 23. Detailed Benchmark Project Manager's Checklist Section 16

Primary Function	**Secondary Function**	**Tasks**
Database Code Loaded on System		
S	C	Install VSD code, Performance Toolbox and other tools.
C	P	Install database if available.
C		Mount database on each node if needed.
C	P	Create benchmark log file.

Table 24. Detailed Benchmark Project Manager's Checklist Section 17

Primary Function	Secondary Function	Tasks
Team Arrives at Benchmark Center		
CP		Arrival kickoff meeting.
CP		Introductions and function discussed.
C		Explain support structure.
C		Explain support shift schedules.
C		Provide welcome package with benchmark-specific information.
CP		Obtain hotel phone numbers.
C		Provide beeper numbers for weekend coverage (when available).
C		Review workroom assignments.
NC		Warn about limited access to analog lines for modems.
N	C	Network setup and usage guidelines.
SN	C	Review SP system configuration and naming conventions.
CN		Resource review - printers, PROFS access, copiers, FAX, speaker phones, and so on.
C		Cover security concerns.
CP		Project plan review.
CP		Discuss meeting plans and status schedules.

Table 25. Detailed Benchmark Project Manager's Checklist Section 18

Primary Function	Secondary Function	Tasks
Start Benchmark		
PC		Hold periodic status reviews.
PC		Schedule post-mortem to review benchmark execution.

Table 26. Detailed Benchmark Project Manager's Checklist Section 19

Primary Function	Secondary Function	Tasks
Benchmark Post-Mortem		
PD		Review outcome of benchmark and results obtained.
S		Review benchmark log for outages and corrections.
CP	DS	List positive and negative influences on the benchmark and recommendations for improvements.
CP		Obtain customer satisfaction survey form.

Table 27. Detailed Benchmark Project Manager's Checklist Section 20

Primary Function	Secondary Function	Tasks
Benchmark Report		
PA		Document benchmark results in a benchmark report.

B.9 Risk Assessment Form

This risk assessment form is used to determine the chances of running a successful benchmark without major problems.

Many benchmark have show-stoppers (features that are clearly going to make a benchmark impossible). Some benchmarks have a lot of lower-risk features that are not incapacitating by themselves, but if they are added together, they can cause the benchmark to fail. This form should help you identify risky benchmarks before the event. If you find you have a risky benchmark then you have a couple of options:

- Tell the benchmark team, benchmark center manager and the customer that this is a risky benchmark. This risk assessment form can then be used to explain your concerns.
- Work as a team to eliminate as many of the risky items before the benchmark starts.
- Add more contingency time to overcome the problems that will arise.

How to use the risk assessment form:

- Go through each question and circle the answer number.

- If the answer number is 2 or 3 you can use a lower number if you are confident that the item poses a lower risk to your benchmark.
- Once completed, add the answer numbers up and compare it to the scoring table that follows.

Note: Questions marked ** pertain to potentially incapacitating situations.

Table 28. Risk Assessment Form Section 1

No.	Question	Yes	No
Database and Application Section			
1	Does the customer application use a database?	0	0
2	Is the RDBMS version Generally Available (GA)?	0	2
3	Is the RDBMS version supported on the version of AIX to be benchmarked?	0	2
4	Is this a parallel RDBMS?	2	0
5	Does the RDBMS have multi-threaded support (SMP or MP)?	0	1
6	Has the customer application been tested on the RDBMS version to be benchmarked?	0	1
7	Has this version of the application been proven in a production environment?	0	1
8	Has this application been tested on AIX?	0	2
9	Has the application been tested on the version of AIX to be benchmarked?	0	1
10	Will the customer install, configure and test the application for the benchmark?	0	3 **
11	Is the database data taken from another system (that is, not generated)?	0	1
12	Is the data in a state ready to benchmark (that is, no data preparation/bulking up work required)?	0	1
13	Will the customer undertake all work relating to data generation and preparation for the purpose of the benchmark?	0	2
14	Is this a client/server application?	1	0
15	Is this an X Windows application?	1	0
16	Is this a character-based application?	0	1
17	Is this a graphical PC-based application?	2	0

Table 29. Risk Assessment Form Section 2

No.	**Question**	**Yes**	**No**
Third-Party Products (excluding a RDBMS)			
1	Are they any third-party products to be used during the benchmark?	1	0
2	Will the customer supply their own copy of the software, including all necessary licence keys?	0	1
3	Is the customer responsible for installing, configuring and testing these products?	0	2
4	Are the product versions supported on the version of AIX to be benchmarked?	0	3 **
5	Is all required data in a state ready to benchmark (that is, no data generation/preparation work required)?	0	1
6	Will the customer undertake all work relating to data generation and preparation?	0	2

Table 30. Risk Assessment Form Section 3

No.	Question	Yes	No
Operating System and Hardware			
1	Is the AIX version Generally Available (GA)?	0	2
2	Is NFS or NIS used during the benchmark?	1	0
3	Is HACMP used during the benchmark?	2	0
4	Are system management products used (like NetView/Tivoli)?	2	0
5	Is the RS/6000 model for the system under test GA?	0	2
6	Has the combination of AIX version and RS/6000 model been benchmark tested before?	0	2
7	Are there any non-IBM hardware components used (such as disks, peripherals, PCs, servers)?	1	0
8	If so, is the customer responsible for providing, installing and testing such devices?	0	2
9	Are SSA disks to be used for the benchmark?	0	1
10	Is RAID to be used for the benchmark?	2	0
11	Is Ethernet (10 Mb), token-ring or FDDI used for the network?	0	1
12	Are there other networks to be used?	2	0
13	If so, is the customer responsible for providing, installing and configuring such a network?	0	1
14	Other than TCP/IP, is there any other network protocol (such as SNA, IPX/SPX, LAT)?	2	0
15	If so, is the customer responsible for providing, installing and configuring such a network protocol?	0	2

Table 31. Risk Assessment Form Section 4

No.	Question	Yes	No
Technical and Benchmark Skills			
1	Will a member of the benchmark center manage the benchmark?	0	1
2	If not, has the nominated person managed a benchmark before?	0	2
3	Is the customer familiar with the benchmarking process (that is, done a benchmark before)?	0	1
4	Has the customer benchmarked this application before?	0	1
5	Will the customer be able to attend planning meetings prior to the benchmark?	0	3 **
6	Will the customer be present during the entire benchmark?	0	2
7	Will the customer representative be able to provide application and database technical assistance for their application?	0	3 **
8	Will a database specialist be required from IBM to assist the data generation or application building?	1	0

Table 32. Risk Assessment Form Section 5

No.	Question	Yes	No
User Simulation Software (PreVue here means any similar tool familiar to the benchmark center)			
1	Will PreVue (dumb screen character-based) product be used?	0	0
2	Will PreVue X Windows be used?	1	0
3	Will PreVue client/server be used?	2	0
4	Has the customer used this tool or are they fully-familiar with its functions and restrictions?	0	2
5	If so, do they have scripts which could be used (this is not possible if the application has changed since)?	0	1
6	Does the customer have fully-detailed lists of the application screens and user input for the benchmark parts of the applications?	0	2
7	Will a load testing mechanism other than the above be used (for example customers own tool)?	2	0
8	If so, will the customer provide software and technical skills to facilitate the benchmark?	0	2
9	Does the number of users to be simulated exceed 512?	1	0

Once you have added up the score for your benchmark, you can use the table below to determine how risky the benchmark is.

Table 33. Risk Accessment Lookup Table

Risk Category	Points Scored	Brief Description
Low	0 - 9	Typical simple dumb screen application benchmark within a known environment
Medium	10 - 22	Out of the ordinary, but should not present problems beyond the skills of the benchmark specialists given sufficient time.
High	23 - 44	An unknown entity. There are elements outside the control of the benchmark center, or there are unproven environments. Such issues require careful consideration and planning. Take a cautious stance and build in caveats and contingency during planning.
Impossible	44 - 88	Stop and rethink the benchmark.

B.10 Sample Benchmark Feedback Form

The following is a sample benchmark feedback form.

To help us in evaluating the effectiveness of our benchmarking program, and to ensure it meets the needs of future participants, we would be grateful for your comments on this sheet. Thank you.

Your Name	
Company Name	
Benchmark Date	

Please circle your selection. **Note:** 1=poor and 10=excellent.

1. How would you rate the benchmark hardware and software facilities?

1	2	3	4	5	6	7	8	9	10

- Is there anything else that you would like included:

....

2. How would you rate the benchmark center technical support personnel?

1	2	3	4	5	6	7	8	9	10

- Is there anything else they could have done?

....

3. Did the benchmark determine all the performance areas you expected to address?

1	2	3	4	5	6	7	8	9	10

- If not, what was omitted?

....

4. Were you happy with the performance results?

1	2	3	4	5	6	7	8	9	10

- Any comments?

....

5. How would you rate your overall satisfaction with the benchmark?

1	2	3	4	5	6	7	8	9	10

- Any comments?

....

6. What was the best/worst aspect about the benchmarking center?

- Best:

  ....

- Worst:

  ....

7. How would you rate the benchmark center against competitors' centers?

1	2	3	4	5	6	7	8	9	10

   - Which benchmarking centers have you worked with before? How are they different?

     ....

8. What is the likelihood that you will now install (further) AIX systems?

1	2	3	4	5	6	7	8	9	10

9. Are there any suggestions or comments you would like to make regarding any aspect of the benchmark center?

   ....

   ....

   ....

Thank you for your assistance with quality measurement of the benchmark center.

Please return this form to any member of the benchmark center.

B.11 Results Presentation Checklist

The results presentation checklist is largely the same for any good presentation:

- Do you know the audience?

- Are the key decision makers present?
- What are the important areas for the audience?
- What are the good points about the benchmark results?
- What are the bad points and do we have a good explanation?
- What are the conclusions to be stressed?
- Can you make the business case for the customer to proceed?
- Are there further actions to be completed, by whom and by when?
- Have you prepared enough for the presentation?

B.12 Success Story Checklist

This is used to maximize the benefits of the knowledge gained on the benchmark. This should include:

- Customer name, if allowed.
- Customer's type of business.
- Time and value of the sale.
- The type of system involved, for example OLTP, DSS, network computing plus accounting, telesales, customer support, human resources, and so on.
- Brief details of the benchmark configuration, such as the system type, memory and disk size.
- Brief details of the results, such as number of users or the batch throughput.
- Compare the above to the success criteria.

If the benchmark was head-to-head against the competition, then also include:

- How much the competition was behind, if known.
- Why your machine was better, using a statement such as the following:
  - Our machine has more powerful CPUs, the disk subsystem gives us a NN% advantage, particular applications are well-tuned or suited for our platform.
- Highlight any know weaknesses in the competition, for example, any reliability issues (did their system crash) or connectivity issues.
- Statement describing how your benchmark team skills are superior.

If particular techniques made large difference in performance, and only if the audience has technical interests:

- Briefly describe the technique and its effect, but state where more information can be found on a more permanent place (usually a Web site).

B.13 Checklist for Being a Good Benchmark Host

Benchmarks can be stressful and involve long hours. The following list helps to make sure customers can get to the benchmark center and concentrate on the benchmark, rather than administration.

Does the customer have the following before joining the benchmark?

- Complete name, address and telephone number of the host(s) and host location.
- A map of how to find the benchmark center.
- Personal contact of whom to meet, when to meet them, and a backup name.
- List of all travel arrangements to the center.
- Complete set of documentation, including the benchmark plan, workplan and responsibilities.
- List of hotels and their telephone numbers in the area.
- List of good restaurants with their addresses and telephone numbers.
- Details of working hours.
- Understand if the following items are available:
  - Telephone with local, national and international access.
  - FAX incoming and outgoing.
  - Photocopier.
  - Dial out access for a laptop personal computer.
  - Mains power rating, for international visitors.
- Details about any special diet or access options available at the benchmark center.

Appendix C. Tools Used in Benchmarking

This appendix has brief descriptions of the tools that are often used by benchmark people. Many people have their own favorite tools and might disagree with this list, but it is a good start. There are hundreds of tools that could be used for performance tuning AIX systems, but the following are the most-widely used.

Note

This Appendix is highly RS/6000 and AIX-specific.

We have not included a detailed list of the tools' functions and features, because you are expected to be familiar with them. We do say why each tool is used and mention its strong points.

For more information about these tools see:

- The technical manual pages or InfoExplorer.
- *Understanding IBM RS/6000 Performance and Sizing* (SG24-4810)
- *AIX Performance Tuning Guide* (SC23-2365)

C.1 General Tools

The following is a list of tools that are useful in the benchmark process.

- A **spreadsheet**.

  There is a Lotus 1-2-3 spreadsheet that details most past and present RS/6000 models in terms of their performance characteristics, called fastsize. This is only available internally to IBM and to business partners. This is used to size and compare RS/6000 machines.

  A spreadsheet is commonly used for analyzing the captured benchmark results and drawing graphs of machine utilization. PTX (see below) and other tools can output spreadsheet-loadable data. This is useful for the analysis of results and for writing the benchmark report.

- A **word processor** for writing the benchmark report.
- The **IBM RS/6000 Configurator** for comparing the costs of various RS/6000 models and exploring configuration alternatives. For more information contact your local IBM representative.
- **Best/1** is a third-party tool from BGS Inc. that provides capacity planning for a range of machines, including RS/6000. This can be used

to in place of a benchmark, by capturing data on a running system and using the data to predict alternative workloads and configurations. It can also be used during a benchmark to predict the results on alternative hardware if a running system is not available for capturing the system. For more information see the BGS Inc. Web page at `http://www.bgs.com`.

- **PreVue** and **Performix** are third-party tools from Rational Inc. These are user workload generators. They can also be used for functional regression testing on new products. During benchmarks they are used to emulate users in an accurate and reproducible way. There are other similar tools, but these are currently the most-widely used in USA and EMEA-based benchmark centers. For more information see the Rational Inc. Web page at `http://www.rational.com`.

C.2 Technical Tools

These tools are used to analyze the benchmark run while it is happening. They are common tools and should all be available on a benchmarking machine.

Note: Some tools require additions LPPs to be loaded.

For more detailed information about all the tools listed in this section, refer to *Understanding IBM RS/6000 Performance and Sizing* (SG24-4810) or the *AIX Performance Tuning Guide* (SC23-2365).

- **Performance Toolbox for AIX** (PTX)

  This tool is a major competitive advantage. It is very good for watching what the machine is doing and includes the time dimension. It has a graphical interface that makes it simple to configure and interpret.

  Particularly useful functions include CPU percentages for each SMP CPU, disk busy percentage, physical and virtual memory and network traffic.

  PTX does need some configuration, so save your favorite monitor once it is set up.

  One 19-inch graphics screen should be permanently allocated to this for each benchmark.

  – **Note:** PTX is not currently part of the base AIX system but comes on separately available CD-ROM. Also note that it is in two parts, the *client* code, which provides the graphical screen rendering and consequently consumes many CPU cycles. Therefore, it must not run on the system under test machines. The *agent* code that runs on the test system provides statistics to the client machine and sends its data by TCP/IP.

- **rmss**

  This tool is used to effectively remove available memory. This is used in benchmarks to try benchmark test runs with different amounts of memory without having to actually physically remove it from the machine. This saves time, wear and tear on the machine and eliminates the possibility of creating a hardware fault during testing.

- **vmstat**

  A simple, tool but effective for a quick look to see CPU usage, memory utilization and paging.

  – **Note:** `vmstat` does not include any time stamps in the output, so it is not good for benchmark report writing.

- **iostat**

  An effective tool to monitor disk activity and spot high usage disks.

  – **Note:** It does not include any time stamps in the output, so it is not that good for benchmark report writing.

- **smitty**

  You might be wondering why we have mentioned this but a lot of administration needs to be quickly and accurately executed. Every benchmark person uses `smitty` for this and never the other alternatives.

- **filemon**

  This is the tool to investigate which file on a disk or file system is the cause of high disk activity (or hot spot). It gives sufficient details to help improve performance in applications that read and write files.

- **sar**

  Similar to `vmstat` and `iostat` but gives more information. You need some AIX kernel internals knowledge to interpret the output.

  – **Note:** On AIX it does not output disk statistics.

- **prof** and **tprof**

  These are very good if you have source code for the application and need to track down the hot spots in the applications algorithms.

- **netstat**

  A simple tool that can help track down network problems and saturation.

- **svmon**

This is the best and only tool for investigating memory usage.

C.3 Advanced Tuning Tools

These tools actually change the way AIX works and are dangerous in the wrong hands. Only use them if you are confident that you understand what you are doing. The new settings should be included in the benchmark report.

- **vmtune**
- **schedtune**
- **bindprocessor**

See the redbook: *Understanding IBM RS/6000 Performance and Sizing* (SG24-4810) for more information.

C.4 Publicly-Available Tools

These tools are not supported by IBM and no warranty is given or implied by including these tools in this redbook.

- **monitor**

  This puts all of the performance-related statistics on a dumb screen terminal in a readable format and continually updates it. It includes the information available in `iostat`, `vmstat`, `sar` and `netstat`, such as, CPU (including SMP), memory, disks, NFS, network, system calls and top processes, CPU, memory use, priority and status.

  This free tool was written by Jussi Maki (jmaki@csc.fi) and the source code can be found through the internet at `http://www.csc.fi/jmaki/` and it is also available internally within IBM.

- **nmon**

  This tool reports most of the data available in monitor, but to a file that can be read by a spreadsheet.

  This is available internally to IBM and business partners at `http://w3.aixncc.uk.ibm.com/tools/aixtools.html`.

- **GNU**

  Don't forget this as a useful source of compilers, tools, algorithms and sorting methods that come with source code. It is also as machine-independent as any tool currently available. GNU is available from the Internet.

  The authoritative FTP site is: `ftp://prep.ai.mit.edu/pub/gnu/`.

A list of FTP sites that have GNU software can be found under: `http://www.cs.pdx.edu/trent/gnu/sites.html`

Use one of the these geographically-appropriate FTP sites for faster downloads.

- **wit**

  This is a korn shell script that outputs the configuration of a machine including the CPU, disks, logical volumes, file systems, memory and adapters in a simple, readable format. This can be used in benchmark reports.

  This is available internally to IBM and business partners at `http://w3.aixncc.uk.ibm.com/tools/aixtools.html`.

Appendix D. Industry Standard Benchmarks

This appendix looks at a few of the many industry standard benchmarks that are available.

D.1 Most-Relevant Industry Standard Benchmarks

For sizing and benchmarks, the most popular and relevant industry standard benchmarks are:

- **SPEC**
  This is a group of relatively small tests. This includes CPU tests for non-floating-point instructions, tests for SMP-based machines and tests for floating-point performance. There are different versions (based on the year the standard was set) that cannot be compared with each other. These tests only give results for the raw CPU performance since they do not, for example, require disk I/O. For more information on the SPEC benchmarks and results, see the following Web site, at `http://www.specbench.org`.
- The Transaction Processing Council (TPC) created two benchmarks that simulate production system. For more information on the TPC, their benchmarks and the results, check the following Web site: `http://www.tpc.org`
  - **TPC-C**
    This is an online transaction processing benchmark using a banking model. The application has just transaction but only two of these transactions make up 88 percent of the workload. There are only five tables in the database. TPC-C particularly highlights I/O and memory bottlenecks. For large configurations, this usually involves a database server with many smaller application servers and a fast network. The application code is very small. The results include the performance and also a price/performance figure.
  - **TPC-D**
    This is a Decision Support System (DSS) benchmark. This test has a specified database schema of eight tables with 73 percent of the data in one large table. There are seventeen specified SQL queries. The database has to be one of a number of fixed sizes and usually above 100 GB. The benchmark includes a data generator program. The results are heavily-dependent on the RDBMS optimized query plan and parallelization. The results include the performance and price performance figures but results against different sized databases cannot be compared.

These benchmarks are formally audited and published on the Web.

There are also developing industry benchmarks for most applications, such as:

- Web servers
- Lotus Notes server
- SAP
- BAAN
- PeopleSoft

These are sometimes written by standards bodies or a particular application provider to allow comparison between platforms.

Industry standard benchmarks are designed to highlight the differences in performance between machines rather than the performance of a production system. Also, note that many of these benchmarks are essentially simple approximations of production systems, and the customer's real database and applications are going to be much more complex. This means that Industry standard benchmarks are not going to be a lot of help in deciding what performance a customer is likely to achieve, and it is dangerous to try to use them for sizing production workloads. If the customer workload is very similar to one of these industry standard benchmarks, it should only be used to make a first approximation of the configuration size.

D.2 Interesting Things about Standard Industry Benchmarks

For the TPC benchmarks, the hardware vendor and RDBMS vendor usually work in close collaboration to run a benchmark. Typically, two to four people (plus permanent RDBMS vendor people) dedicate six months to a single benchmark result. These specialist do nothing but run and tune the same benchmark repeatedly. They become highly-skilled in one benchmark and specialized in every option and technique for gaining performance.These specialist tell us that seemingly minor changes in configuration or setup can make large differences in results.

But we have to be careful about these results because they do not represent real life. The workload is very limited (production systems have varying workloads) and most production systems cannot afford the tuning manpower applied to these benchmarks. The specialists that do these benchmarks told us some interesting things, as listed below.

- **SPEC** compiler options can make enormous differences. The standard settings are good for most applications but the SPEC settings may actually make performance worse for normal applications.

- **TPC-C** uses more disks and larger memory sizes than a sensible production system, to ensure that the only bottleneck left is the CPU.
- **TPC-D** has the most understood SQL statements; hundreds of man hours have been used analyzing every possible option and alternative.
- For the database test, no special tuning tricks are used, only the standard RDBMS parameters or options settings.
- Some tests allow the use of products that are not released, but will be in the future.
- Benchmark figures improve without *any* hardware changes due to improved test application software. Over the years the path length reduces for industry standard test applications, but in production systems, the path length typically increases as the system evolves and matures.
  Note: Path length is the number of CPU instructions a particular transaction or work unit requires.

D.3 Car Racing Metaphor

A metaphor using Formula 1 or racing cars to compare industry standard benchmarks and production systems is useful, as listed below.

1. These racing cars are highly-tuned and dedicated to the highest possible performance.
2. A single racing car has a dedicated team supporting and tuning it.
3. These racing cars do not represent road cars (such as your family sedan).
4. These racing cars do not tell us how fast a road car will go.
5. The results of the last race are not a good indication of which road car to purchase next.

The same things can be said of industry standard benchmarks:

1. These benchmarks are highly tuned and dedicated to the highest possible performance.
2. A single benchmark has a dedicated team supporting and tuning it.
3. These benchmarks do not represent production systems.
4. These benchmarks do not tell us how fast a production system will go.
5. The results of the last benchmark are not a good indication of which production hardware to purchase next.

D.4 The Reasons for Industry Standard Benchmarks

If industry standard benchmarks are so unlike production systems; why are industry standard benchmarks run?

- They allow customers and vendors to compare and size products.
- They allow research on RDBMS, operating systems and hardware. This research will eventually produce better products.

One exception is the TPC-D benchmark. This comes with a well-documented schema (eight simple tables that are typical for a simple order processing database) and a very useful, scalable data generator. Many benchmark people have used this database as a research and learning tool since it is easy to build and populate. It can also serve as the starting point for developing a new benchmark test, so you do not have to reinvent the wheel. However, only the formally audited and reported results can be claimed as TPC-D, and we must not expect to get performance similar to the formal TPC-D results (unless we apply the same amount of time and effort, which is clearly impractical in customer benchmarks).

Appendix E. Special Notices

This publication is intended to help project leaders, project managers and consultants plan a benchmarking project. This redbook is also intended for technical specialists, technical marketing representatives, and system engineers who are involved in the execution of benchmark tests and the presentation of their results. See the PUBLICATIONS section of the IBM Programming Announcement for more information about publications that are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, NY 10594 USA.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (″vendor″) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each

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

AIX®	CICS®
DB2®	IBM®
InfoExplorer	Lakes
MVS® (logo)	PROFS®
RISC System/6000®	RS/6000
SP	

Appendix F. Related Publications

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

F.1 International Technical Support Organization Publications

For information on ordering these ITSO publications see "How to Get ITSO Redbooks" on page 165.

- *RS/6000 Performance Tools in Focus*, SG24-4989
- *Understanding IBM RS/6000 Performance and Sizing*, SG24-4810
- *Customizing Performance Toolbox and Performance Toolbox Parallel Extensions for AIX*, SG24-2011

F.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
Lotus Redbooks Collection	SBOF-6899	SK2T-8039
Tivoli Redbooks Collection	SBOF-6898	SK2T-8044
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
RS/6000 Redbooks Collection (PDF Format)	SBOF-8700	SK2T-8043
Application Development Redbooks Collection	SBOF-7290	SK2T-8037

F.3 Other Publications

These publications are also relevant as further information sources:

- *Performance Tuning Guide*, SC23-2365
- *Performance Toolbox for AIX: Guide and Reference*, SC23-2625
- *Performance Toolbox Parallel Extensions for AIX: Guide and Reference*, SC23-3997

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 `http://www.redbooks.ibm.com`.

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 BookManager BOOKs of redbooks, type the following command:

  ```
  TOOLCAT REDBOOKS
  ```

  To get lists of redbooks, type one of the following commands:

  ```
  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, type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ITSOREGI 1998
  ```

  For a list of product area specialists in the ITSO: type the following command:

  ```
  TOOLS SENDTO WTSCPOK TOOLS ZDISK GET ORGCARD PACKAGE
  ```

- **Redbooks Web Site on the World Wide Web**

  `http://w3.itso.ibm.com/redbooks`

- **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.

- **REDBOOKS 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** — 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:	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 29570	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	1-403-267-4455
Outside North America	(+45) 48 14 2207 (long distance charge)

- **1-800-IBM-4FAX (United States)** or **(+1)001-408-256-5422 (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 Web Site	http://www.redbooks.ibm.com
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

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.

List of Abbreviations

ADSM	automated data storage management
AIX	advanced interactive executive
APA	all points addressable
CICS	customer information control system
CPU	Central Processing Unit
DASD	direct access storage device
DBA	database access
DDL	database definition language
DSS	Decision Support System
EMEA	Europe/Middle East/Africa
FTP	file transfer protocol
GA	Generally Available
GUI	graphical user interface
HACMP	high availability cluster multi-processing
I/O	input/output
IBM	International Business Machines Corporation
ICA	intercompany agreement
IP	Internet Protocol
IPX	Internetwork Packet eXchange
IT	information technology
ITSO	International Technical Support Organization
JCL	job control language
JFS	Journaled File System
LAT	local area transport
LPP	Licensed Program Product
MPP	massively parallel processor
MVS	multiple virtual storage
NFS	network file system
NIS	network information system
NOSS	national office support service
OLTP	online transaction processing
OMSYS	Opportunity Management System
OS	Operating System
PROFS	Professional Office System
PTX	Performance Toolbox for AIX
RAM	random access memory
RDBMS	Relational Database Management System
RS/6000	IBM RISC System/6000
S&E	Scientific and Engineering
SAP	Systems, Applications, Products in Data Processing
SMP	symmetric multiprocessor
SNA	systems network architecture
SP	Scalable POWERParallel
SPX	Sequenced Packet eXchange
SQL	structured query language
SUT	system under test
TCP/IP	Transmission Control Protocol/Internet Protocol
TP	transaction processing
TPC	Transaction Processing Council
VMM	Virtual Memory Manager
VSD	virtual shared disk

Index

ITSO Redbook Evaluation

Benchmarking in Focus
SG24-5052-00

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.ibm.com
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to redbook@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!)**

____

____

____

____

____

SG24-5052-00
Printed in the U.S.A.

Benchmarking in Focus

SG24-5052-00