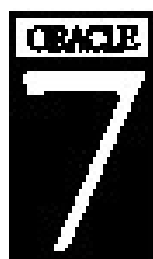


Oracle7 Sizing and Capacity Planning Guide for VAX OpenVMS Systems

MAY 1994



Alpha AXP

**21ST CENTURY
COOPERATIVE COMPUTING
PEOPLE & DIGITAL**

**Oracle7 Sizing and Capacity Planning Guide
for VAX OpenVMS Systems**

Part No. A19176

May 1994

Author: Scott Foote

**Contributors: Lakshmi Arya-Bakshi, Judy Clifford,
Jennifer Stave**

**Copyright © Oracle Corporation 1994
All rights reserved. Printed in the U.S.A.**

Restricted Rights Legend

**Use, duplication, or disclosure is subject to restrictions
stated in your contract with Oracle Corporation.**

**Use, duplication, or disclosure by the
Government is subject to restrictions for commercial
computer software and shall be deemed Restricted
Rights software under Federal Law.**

**The information in this document is subject to change
without notice. If you find any problems in the docu-
ment, please report them to us in writing. Oracle
Corporation does not warrant that this document is
free of errors.**

Oracle7 is a trademark of Oracle Corporation.

**Lotus 1-2-3 is a registered trademark of Lotus
Corporation. .**

**OpenVMS, VAX and Alpha AXP are trademarks of
Digital Equipment Corporation. Corporation.**

- Before making a purchase decision, a prospective buyer wondering whether to invest in new technology will inevitably ask the salesperson how much performance can be gained from that investment. “How well does your software perform on my system?” is one of the most important questions asked during the software sales cycle, and often one of the hardest to answer.

Application performance depends on the capacity of the customer’s hardware as much as it depends on the capabilities of the vendor’s software. How “your software” performs on “my system” also depends on how often and how well the system is tuned, what kind of workload the application must support, and size and structure of the database.

The prospect and the salesperson should always work together to define not only a software solution, but also the CPU, memory, and disk resources needed to meet the prospect’s performance expectations. This book describes an effective, reasonably simple method of using a customer’s business needs and application requirements to estimate these resource requirements.



I. Introduction

Purpose

This guide describes a simple but effective method for estimating the CPU, main memory and disk I/O requirements of applications based on Oracle7 . This guide focuses on estimating CPU and memory requirements, but also explains how to estimate the required number (not the size) of disk drives based on the I/O load (or “rate”).

Estimating CPU and Memory Requirements

Calculating memory requirements is less complex than calculating CPU requirements. Once a user activates a tool and connects to a database, the amount of physical memory consumed by the user process is fairly constant (not considering paging, swapping, and large sorts), whether or not that user is actually doing any work. However, the amount of CPU consumed by the user process will vary greatly depending on the level of user activity.

So, rules of thumb based on empirical data are reasonably accurate methods of quickly estimating *memory* consumption. But using rules of thumb to estimate *CPU* consumption is not recommended.

Estimating Disk Requirements

The amount of disk I/O an application will generate also depends on the level of user activity. However, I/O rates also depend on the size and physical structure of the database, the size of the SGA and the level of SQL code optimization. Since you do not generally have such information when *sizing* an application, this guide includes a quick method for estimating the peak I/O rate and the number of disk drives needed to support that I/O load.

The results of any sizing or capacity planning process are only as accurate as the data on which the process is based. Therefore, Appendix A includes a detailed example of the types of questions you can ask to gather this information from your customer or prospect.

Contents

This guide shows you how to complete sizing and capacity planning tasks on paper. However, it also includes the mathematical calculations and the sizing estimate tables in a Lotus 1-2-3 spreadsheet. Appendix D contains instructions for using this spreadsheet.

The remainder of the document is organized as follows:

Part I, “*Before You Begin*,” defines some basic concepts and explains several preparatory steps you should complete, including how to tune a production Oracle RDBMS environment to achieve optimal performance, and identifies several areas to consider when tuning.

Part II, “*What to Expect*,” reviews the general method for sizing and capacity planning. This method involves four basic steps:

- Step 1. **Define** customer requirements.

When sizing, collect information from the customer about how the planned application will be used. This information will help you define the throughput (or workload) requirements. You should complete this step using the customer survey in Appendix A. When planning capacity, determine what your customer considers acceptable response time for the existing application and the expected user workload.

- Step 2. **Measure/Estimate** CPU and memory use.

When planning capacity, you can measure the resources used by testing the existing application under its current workload and extrapolate from your test data what resources are needed to support a larger workload. When sizing, you must estimate the resources that will be consumed because the application does not yet exist. To estimate resources, use the data you gathered in Step 1 and the information in the sizing tables included in Part III.

- Step 3. **Predict** total CPU and memory requirements.

Use the projected number of users and their expected levels of activity to estimate the system resources that a planned application will require in its production environment, or that an existing application will require to support a larger workload.

- Step 4. **Model** required CPU capacity.

Use the CPU requirements estimated in Step 3 (expressed in terms of total response time) in a queuing model that determines the CPU capacity required to support the application, support the CPU load, and ensure acceptable performance.

Part III, “*The Sizing/Capacity Planning Process*,” describes the step-by-step procedure for implementing the four basic steps described above for both sizing and capacity planning exercises.

Part IV, “*Useful Reference Information and Tools*,” contains several appendices of reference information that will help you collect and analyze information during the sizing or capacity planning process.

Part I. Before You Begin

Step 1. Learn the Basic Concepts

In this step, you will learn what the basic goal of sizing and capacity planning is and what information you'll need to reach this goal.

What are Sizing and Capacity Planning?

Sizing is the process of estimating the hardware requirements of a planned application, based on technical descriptions of your customer's business needs. The more detailed and accurate these descriptions are, the more accurate your sizing estimates will be.

You will be able to estimate effectively the system capacity needed to support the end-user application only after you and your customer:

- clearly define the end-user application (or at least determine the business needs it must handle)
- define the expected database activity (number and type of database transactions, number/type of users, etc.).

This guide provides several useful tools for collecting this information, including a customer survey and examples of CPU and memory use by common transaction types.

Capacity planning is the process of measuring resource requirements from existing applications, and projecting the amount and kind of hardware required to support larger workloads. Successful capacity planning also depends on detailed, accurate information, but requires *measuring*, rather than *estimating*, application requirements.

What Is My Goal?

Whether sizing or planning capacity, your goal is to define the minimum hardware needed to meet your customer's performance requirements. In both processes, you will attempt to assign values to several key variables, which are discussed in detail later in this guide:

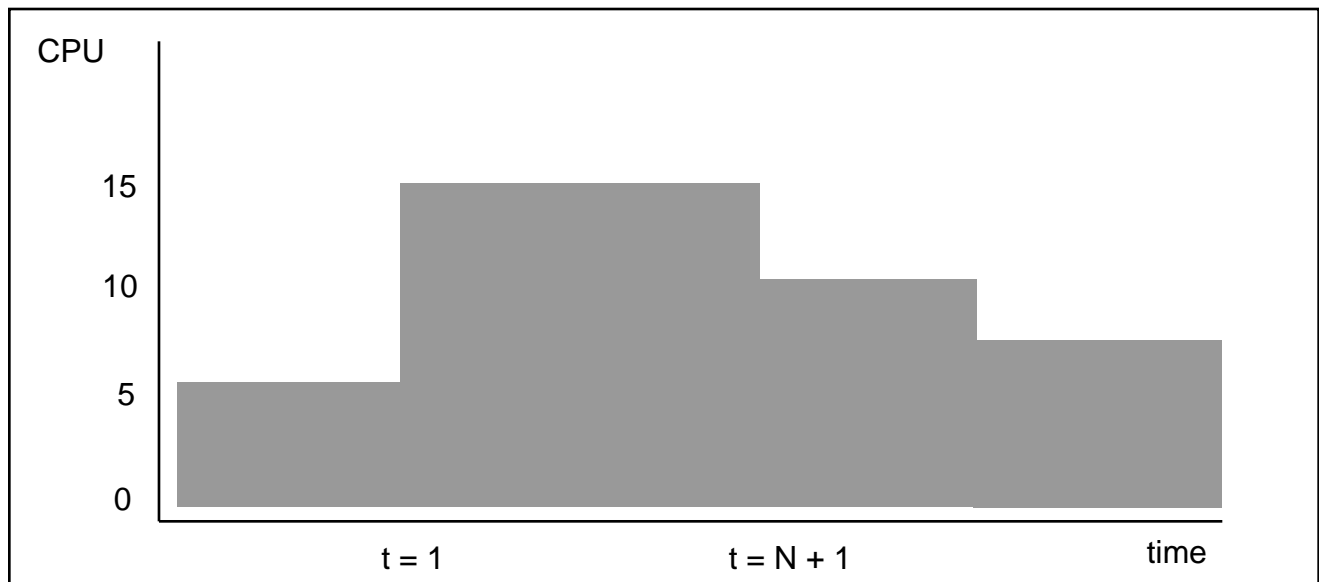
| | |
|----------------------|---|
| V | is the minimum uniprocessor rating your customer requires. The value of V determines which system (V_c) is appropriate for your customer's application. |
| M_t | is the estimated memory needed to support the application users (not including the SGA) using your customer's applications. |
| D | is the number of disks needed to provide optimal performance on your customer's system. |
| X | is the estimated impact of the system you recommend (V_c , c , and D) on the application's desired response times. |

More about Sizing

In new sales situations, you must usually base your sizing estimates on the projected workload of non-existent applications. But the Oracle RDBMS is not an end-user application; the Oracle RDBMS is a database system which an end-user application uses to manage information stored on disk.

Therefore, estimating the hardware capacity required to support a specific number of "users running the Oracle RDBMS" is difficult, because the users do not themselves run the Oracle RDBMS. These users instead run an application which uses the Oracle RDBMS. So, the hardware required to run an Oracle RDBMS-based application depends, in part, on the degree to which the end-user application uses the Oracle RDBMS .

Figure 1. CPU Used in Relation to Time



What is CPU Workload, and How Do I Measure It?

CPU workload is measured by comparing the amount of CPU capacity used to the amount of time the CPU is under load. The capacity of the Digital VAX computers discussed in this guide is measured in VUP units. A VUP is a VAX Unit of Performance that represents CPU capacity relative to a VAX 11/780 system. Therefore, workload on a VAX is measured as number of VUPs per second.

For example, a transaction that generates a workload of 20 *VUP-seconds* will require 20 seconds of CPU time on a one-VUP VAX (i.e., the 11/780). The same transaction will require only four seconds of CPU time on a five-VUP VAX. The workload does not change between the two scenarios; increasing CPU capacity is responsible for reducing the CPU time that the transaction requires. So, workload is not CPU-specific.

Figure 1 illustrates use of CPU resource in relation to time. CPU capacity is measured in VUP units and time is measured in seconds. So, during the work cycle between $t=1$ and $t=N+1$, the total workload the CPU supports is:

$$15VUPs * (N+1-1) \text{ seconds} = 15N \text{ VUP-seconds}$$

Step 2. Invest in Tuning.

Before you begin analyzing the requirements for your customer's planned or existing application, consider the current state of the environment on which the application will run or does run. Possibly the most significant and least considered factor in sizing or capacity planning is the performance returned from tuning the system, which includes the application itself, the database, and the operating system.

All information systems require regular analysis and maintenance to continue providing optimum performance. Regularly tuning an Oracle RDBMS database and the applications which use it is necessary to maintain an information system, just as preventive maintenance is necessary to maintain the actual hardware components.

Unless a user expects database information to remain completely static (no updates, no inserts, no deletes, etc.) and the user workload to remain constant, no amount of initial tuning will suffice for the life of the information system. Refer to *VAX/VMS Tuning for ORACLE* White Paper (part number 52569-0992) for detailed information on tuning an Oracle/VMS configuration.

Step 2a. Evaluate System Conditions

The sizing/capacity planning method described in this guide assumes the application has been optimally designed and the system has been well-tuned. The application and runtime environment should meet the following criteria.

System

- No system resource bottleneck exists on the system bus.
- The SGA is properly sized, I/O is efficiently distributed, and there is minimal fragmentation or block-chaining.
- The operating system has been tuned.

- Applications are designed to minimize database lock contention.
- The system runs on a single CPU or a shared memory, multi-CPU (SMP) configuration.
- No runaway processes (unanticipated infinite loops) exist.

Memory

- There is negligible paging/swapping.
- The SGA is liberally sized; the cache hit ratio should be no less than .70.

Network

- There are no significant network delays.
- There is no significant client-server overhead.

Workload

- The workload does not widely fluctuate during the peak usage period.
- The entire system load is understood.
- I/O load components will not bottleneck on particular disks at any given time while leaving other disks unused.

When assuming (or ensuring) these conditions are true, you should also consider the state of the system in relation to the work your customer expects to do. Be sure to consider peak performance requirements during sizing or capacity analysis, so you can confirm the system will handle the entire workload range, from light to heavy, expected to occur daily. Most performance problems occur during peak workload intervals. What type of specific requirements should you consider?

CPU Requirements

When estimating the appropriate CPU size, consider whether the following characteristics exist:

- Large sorts done either in memory or on disk
- Heavy amounts of parsing
- Heavy navigation of complex forms
- Mathematical manipulation of row/column.

Memory Requirements

- Paging is usually undesirable, but may be acceptable in some situations for low-workload users. If the system has been well tuned, swapping typically indicates that the workload has reached the user memory limit. Avoid swapping at all costs; never under-configure memory.
- The optimum SGA size depends on the actual workload and amount of data sharing. When estimating system memory, size the SGA generously.

In addition, consider the following factors affecting memory requirements:

- Maximum number of users is typically a factor of memory usage first, I/O second, and CPU last. (The amount of privately-held memory required by users will vary dynamically.)

- size of unshared images held in memory
- amount of shared images held privately
- amount of sorting
- Amount of SQL statement parsing.

Disk Requirements

When estimating the size and number of required disks, we recommend that you use several small-capacity disks, rather than a few larger-capacity disks. Balancing I/O load across several disks is easier than balancing it across just a few disks. Most disks operate optimally at 30-40 I/Os per second with no disk queue. Also, project database growth carefully. Table, index, or rollback segment fragmentation in a tablespace, or file fragmentation in the .DBS file on disk can significantly erode performance.

Dedicate one disk to the Oracle RDBMS executables. If possible, keep the Oracle RDBMS executables and the database files on different disks; image activation and SGA paging can interfere with database file I/O. Dedicate at least one disk to the redo log files. One disk for each redo log file may significantly help performance if ARCHiving is enabled.

Finally, locate redo log files on their own disk, because the random disk head movements caused by accessing database files or activating executable programs can impact the nearly linear head movements caused by the LGWR process writing sequentially to the redo log files.

Throughput Requirements

The maximum number of transactions per second (tps) is usually limited by both the CPU capacity and the aggregate throughput of the disk subsystem (I/O). Configuring enough disks in the beginning can help minimize I/O bottlenecks. Be liberal but realistic when estimating CPU requirements.

Response Time Requirements

I/O is the primary influence on response time. When CPU resource is not a bottleneck, poor response time can result from heavy I/O due to:

- Large disk sorts
- Poor indexing techniques
- Lack of indexes

Simple factors such as the number of rollback segments or the size of the redo log buffer can also dramatically effect response time. Though not directly I/O-related, excessive parsing and reparsing of SQL statements can also impact a user's response time.

Batch Requirements

Batch activity is inherently CPU-intense. When not I/O-intensive, it has a tendency to consume most of the idle CPU resource available, making the system appear to be CPU-bound at 100% utilization for the duration of the batch run. This consumption of idle CPU time will likely decrease overall performance for interactive users, particularly on a uniprocessor system.

On a multi-CPU system, a single batch job can never consume more than one CPU. For example, if the system has five CPUs,

then the total system capacity would be 500%. A single batch job can never consume more than 100% of the available 500% CPU capacity. Two parallel batch jobs can never consume more than 200% of the available CPU capacity, etc. Therefore, batch activity has less impact on interactive user processes when they become computable. So, choose a multi-CPU (SMP) system when the end-user environment will include any intensive batch processing that may interfere with interactive users.

Step 3. Collect Customer Information.

Before you begin the actual sizing/capacity planning process, refer to the customer survey in Appendix A. The questions listed in this survey will help you gather information about your customer's business needs, application design, and the expected application workload that you'll need to complete the sizing/capacity planning process.

You need not obtain answers to all of the survey questions; the survey is intended only as a template for gathering the type of data you need to define the application and expected user activity. However, the level of detail in the information gathered, and the accuracy of your workload/application estimates will greatly affect the accuracy of your sizing or capacity planning process.

Once you've collected this information, organize the information into three areas that will be useful to refer to later.

User Activity

- Define business goals for the applications.
- Identify peak/average numbers of total/concurrent users
- Define expected system throughput.
- Describe transaction types.
- Identify required reporting features.
- Identify typical *worst-case* acceptable response time for all transaction components.
- Identify Oracle RDBMS products to be used in the applications.

Workload

- Classify user groups as light, medium, or heavy.
- Identify the number and size of reports.
- Quantify reporting activity as batch or interactive.
- Estimate transaction sizes and complexity.
- Estimate transaction rates for each class of user and/or each type of transaction.
- Consider CPU-intensive applications, e.g., highly computational, CASE, batch reports, etc.

Growth Capacity

- Determine if additional disk capacity will be necessary later.
- Determine if the application is suited for the Oracle RDBMS Parallel Server.
- Determine if off-loading users to SQL*NET client systems will minimize CPU bottlenecks.

Part II. What To Expect

Once you've completed the preparation described in Steps 1-3 above, you will be ready to begin the sizing/capacity planning process. Part III explains in detail how to estimate CPU and memory resource requirements; this section simply outlines the four basic steps involved in the sizing/capacity planning process.

The Sizing/Capacity Planning Process

Note: Steps 1 and 2 (*) vary somewhat depending on whether you are sizing or planning capacity. Figure 2 below outlines the steps you'll follow to complete either activity.

* Step 1. Define Customer Requirements

When sizing:

Define the business functions and the resulting database transactions that the customer requires the planned application to support. First, decompose specific jobs or business functions into business transactions. Then, group similar business transactions into transaction types, for example, filling out a form, running a report, updating an account, etc.

Next, describe each business transaction type in terms of expected database activity, for example, query, update, or delete activity, number of tables to be accessed, number of rows to be manipulated, validation requirements, etc.

Note: The tool that will be used to implement these database transaction (e.g., SQL*Forms, SQL*Plus in batch, Pro*C routine, etc.) is important in determining database activity.

When planning capacity:

Because the application already exists, you do not need to define business functions and database transactions. However, you should have an accurate method of quantifying the customer's performance expectations for the application, that is, what response time he or she considers acceptable on a per-transaction basis.

* Step 2. Estimate or Measure CPU and Memory Use

When sizing:

Compare your database transaction descriptions to the transaction test data in the sizing tables, **Typical CPU Workloads** and **Typical Memory Requirements**. Calculate your estimates of total CPU and memory requirements based on the ranges of sample transaction data in these tables.

When planning capacity:

Gather CPU and memory statistics by testing the existing application under a sample of the expected user activity. Normalize the CPU resource used for a particular type of transaction to "CPU-units*seconds" by multiplying the CPU seconds used by the uniprocessor CPU rating of the system under test.

Step 3. Predict Total CPU and Memory Requirements

Now determine the total resources — total CPU (V_t) and total memory (M_t) — required to support the transaction throughput and the number of users that your customer expects on the production system. This information will be known as the Transaction Workload profile (TWP). Use the sizing or capacity planning worksheets in this section to calculate these total requirements. During this step, you can also estimate (or measure) the total I/O rates and determine the optimum number of disk drives (D).

Step 4. Model Required CPU Capacity

Use the CPU requirements from Step 3 and your customer's performance expectations (expressed as total response time) to determine the minimum size and number of CPU's needed to support the workload.

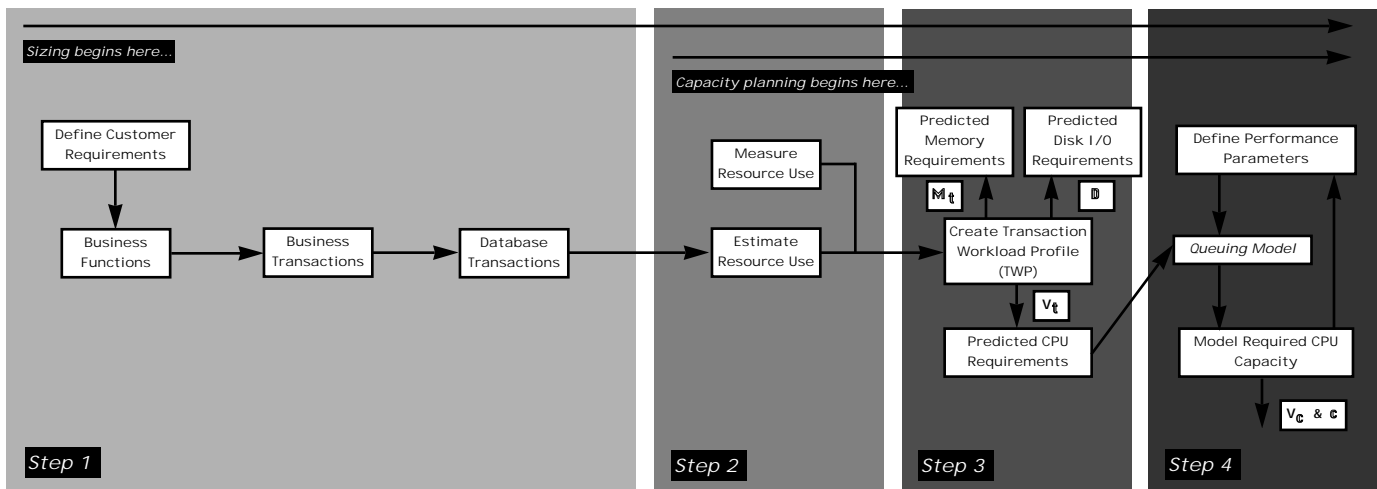


Figure 2. The Sizing/Capacity Planning Process Outlined

Part III. The Sizing/Capacity Planning Process

* Step 1. Define Customer Requirements

Now you're ready to begin. In Step 1, you'll complete three tasks.

- First, define specific jobs (or business functions as business transactions).
- Then group similar business transactions into transaction types. (Examples of business transactions include filling out a form, running a report, updating an account, etc.).
- Next, describe each business transaction type in terms of expected database activity.

**Note that if you are planning capacity, you can omit Step 1, although you may find it useful to group and characterize transaction types in the existing application.*

Step 1a. Define Business Transactions

Interview your customer to form a description of the application's business transactions. A business transaction is any atomic unit within the jobs specified by your customer, for example, entering a new invoice or generating a report. For SQL*Forms-based applications, a transaction is normally defined as the steps ended by a COMMIT KEY, including executing the COMMIT. One criterion for a business transaction is that it can be executed independently of other steps in a job stream, but none of its parts can.

Example

"What business functions or needs will be automated by this application?"

"The primary goal in building this application is to automate the processing of shipping orders from our warehouse. The current process is manual and paper-based, so it's very time consuming and complicates the reporting activity and stock level maintenance. The secondary goal in building this application is to provide up-to-the-minute reports for order tracking and inventory management."

With the customer, break these business functions down into more tangible business transactions. For example, to support the business functionality described above, this application must provide the following:

- A mechanism for operators to enter new orders online, including price calculations and inventory level checking and updating. An order will typically consist of 5-15 line items.
- A mechanism for operators to enter payment received information, including updating the warehouse and district balances.
- A mechanism for operators to track order status.
- A mechanism for warehouse managers to find the current stock level for any item or any group of items.
- A mechanism for warehouse managers to process/deliver one or several customer orders.

Example

"During peak hours of operation, how many concurrent interactive operators and warehouse managers will be actively using the application?"

"During peak operating conditions, there will be up to 30 interactive order entry personnel entering or tracking orders, up to five warehouse managers processing or delivering orders, and up to two administrative personnel running reports against the stock levels and order activity."

"What are the overall volumes of the items and the customers to be handled?"

"The warehouse stocks some 50,000 items and serves approximately 30,000 customers."

"What are the performance criteria for the production environment? How will performance be measured?"

"The primary measure of performance will be response time; meaning the overall elapsed time to perform any function, minus the keying time and any think time. Acceptable worst-case response times for each function are:

| | |
|----------------|------------|
| New Order | 20 seconds |
| Payment | 20 seconds |
| Order Status | 20 seconds |
| Stock Level | 30 seconds |
| Delivery | 90 seconds |
| Admin. Reports | 5 minutes |

"What are typical/minimum structures which might exist in the database?"

"At a minimum the database would need to store information about:

- the customer profile; the stock for each shippable item the item description (i.e., price)
- a list of new orders
- a list of orders being processed
- a table of line items, etc."

Step 1b. Group Transactions into Types

Group similar transactions into types and label the transaction types; for example, "Simple Read-Only Transactions" might be one type of transaction. Before completing this task, you will need to consider several elements:

- First, decide on the time window.
- Next, decide on transaction frequency.
- Allow for load prediction errors.

Decide on the time window

The behavior of most application workloads changes over time. Any period in which the application behavior remains constant is a time window. This can be minutes, hours, days, weeks, months, or any other time period the customer associates with his or her peak usage. For a stable application workload, use a longer window such as an hour or a day. For a very dynamic application workload, use a shorter time window, such as a minute.

Define the peak usage period from the customer's perspective, for example, the week he or she performs quarter close activities. The customer must agree that this time window will be used to distinguish acceptable from unacceptable performance. There may be more than one peak usage period; consider each of them.

Decide on transaction frequency

The frequency of a business transaction is the number of times the transaction occurs in the time window. Normally, you should consider time windows in the busy hours or peak loaded time. If you are studying more than one peak usage period and observe different frequency patterns, consider a time window for each of them.

Example

"How frequently do these business transactions occur?"

"The expected peak rate for new order requests is 350 per 8-hour day. The expected peak rate for incoming payments is 350 per 8-hour day. The expected peak rate for order status requests is 500 per 8-hour day. Each of the interactive operators may perform any of these first three functions. At peak operating conditions, the warehouse managers may request stock level status reports every 30 minutes. The warehouse peak rate for processing the delivery of orders is expected to be 750 orders processed and delivered in a 16 hour day (two shifts). Administrative personnel may also request stock activity or order activity reports at a peak rate of one per hour."

Allow for load prediction errors

Estimating the system resource load is the major source of errors. We recommend that you use a worst-case load approach. Check user inputs to the load estimate process carefully and make certain they are realistic. Estimating loads inaccurately compromises the reliability of all your subsequent predictions.

Step 1c. Translate Business Transactions into Database Transactions.

You should decompose each business transaction, if possible, into database transactions. A database transaction is an atomic database job that is fully completed or fully aborted. Examples of database transactions include a single query, or several updates, or a query and a delete, etc.

Example

New Order

This business transaction can probably be performed within one database transaction that, for each line item, will:

- check inventory levels
- update an order id, sequence number, query rate, and tax information
- calculate price information
- update or insert information describing the new order and the associated line item
- update the stock information.

This cycle will need to be repeated for 5-15 line items. Three to five rows will be processed for each line, an approximate average of 45 rows per transaction. This might be a "medium-weight" transaction compared to others in these applications, because it queries and updates several pieces of information and performs price calculations. This database transaction may need to operate on as many as seven tables.

Payment

This business transaction can also be performed within one database transaction. This transaction will:

- update the warehouse balance
- update the district balance
- update the customer balance and credit rating information.

An average of four rows may be processed for each Payment transaction. This might be a "light-weight" transaction relative to the others because it will involve only a few updates and/or inserts with limited calculations. This database transaction will operate on four or five different tables.

Order Status

The Order Status business transaction can be performed within one simple query-only database transaction that also performs an indexed lookup on customer and order information. This will be a light-weight transaction relative to the others in this application; it will operate on probably one or two tables and process 10 to 12 rows on average.

Stock Level

The Stock Level business transaction may be performed within one or multiple database transactions. Assuming only one query-only transaction is needed, it will require more involved queries for each line item or order being queried. This might be either a medium- or heavy-weight transaction depending upon how it functions. It may perform several nested functions on only one or two tables, but may process up to 200 rows on average.

Delivery

The Delivery business transaction could also be done in only one or in multiple database transactions depending on the expected functionality. Assuming it is done within a single database transaction, the order status will need to be updated as

well as the ship date/time (possibly for each individual line item), the number of items remaining to be shipped may be needed to calculate short-shipped orders; information about the carrier used may also need to be recorded. This database transaction will operate on as many as four tables and probably as many as 250 rows during peak processing periods.

The reporting activity will simply consist of complex queries; possibly containing nested SELECT statements.

* **Step 2. Estimate or Measure CPU and Memory Use**

Now that you have defined the customer's application and workload, you are ready to **estimate** (sizing) or **measure** (capacity planning) how much CPU and memory resource the application end-users will consume.

When sizing:

Estimating these resources accurately is a very complex process. In choosing a method, note that your estimates cannot be more accurate than the method and data you use. We recommend that you use the following guidelines:

- First, avoid using "rules of thumb" to predict CPU usage.
- Use publicized TPC benchmark data only for system comparisons.
- Finally, compare existing test data.

When sizing, keep in mind that these methods can be very misleading because these rules and data probably do not truly represent your customer's unimplemented and often undefined applications. Without completely understanding how an application will use the Oracle database and how heavily end-users will use that application, drawing comparisons to existing data is difficult.

The following sections contain tables of test data that contain the default data you should use in general sizing estimates.

When planning capacity:

You may omit this step and proceed to Step 3. Capacity planning does not require sample test data or rules of thumb because there is usually actual performance data about the existing application to help you characterize its resource requirements. If this information is not available, test the system running under a representative workload to determine CPU, I/O, and memory requirements. Normalize CPU data by multiplying the CPU used (in seconds) by the uniprocessor CPU rating of the system under test.

Avoid using rules of thumb to predict CPU use

Rules of thumb for estimating CPU requirements are very broad generalizations typically based on a diverse range of applications and application workloads. These rules oversimplify the CPU requirements that the mythical "average" Oracle application will require.

Even when the rules incorporate value ranges rather than fixed values, how you should apply these rules to size your "real" application is never clear. For example, even if the estimated range for CPU use is accurate, you must still determine the

point within this range where your application's resource use will fall. If you guess incorrectly, you may over-size the systems and invest in unnecessary expansion, or you may grossly under-size the configuration and fail to meet your customer's needs. Remember that it is always easier to size an application accurately in the beginning than it is to appease an unsatisfied customer later.

Note: Using general rules of thumb to estimate memory requirements is far less risky. Again, this is due to the fact that once a user activates a tool, the amount of physical memory consumed by the user process is fairly constant (ignoring paging and swapping), whether or not that user is doing any work. However, if using a range of memory requirements, you must still determine where the "average" user will fall in this range. Rather than attempt to define this mythical average user, consider different user classes separately, and consider the tools each class will use as well as the complexity of each transaction.

Use publicized TPC benchmarks only to make system comparisons

TPC benchmarks are designed to eliminate much of the overhead of real-world end-user applications. They demonstrate the capacity of CPU needed to handle pure database transaction rates. So, their test workloads typically consist of transactions which use a very small amount of CPU but are processed at very high rates. These benchmarks most resemble high-transaction-volume or OLTP environments.

We often overlook the fact that, in the real world, the end-user application, not the database work needed to process transactions, places the heaviest load on CPU. To arrive at any reasonably accurate sizing estimate, consider the CPU needed to support an application in addition to the CPU needed to support the required database activity.

Always carefully consider application requirements when using benchmark data for sizing or capacity planning. And be sure you understand the transaction types tested by TPC benchmarks before using them. Some of the published TPC Oracle RDBMS benchmarks which may be helpful for reference purposes are listed below. Use this benchmark data for relative comparisons of how well different platforms handle the different TPC applications and workloads. However, unless your customer's application is very similar to the specific TPC benchmark application tested, do not rely exclusively on this benchmark data to size resources.

Refer to the following data sheets:

- *ORACLE Performance Report (V6.0) VAX 6000 Model 500*, Part no 51384-1090
- *ORACLE Performance Report ORACLE V6.2 on the VAX 6500*, Part no. 52336.1291
- *ORACLE Performance Report (V6.2) ORACLE Parallel Server for VAXcluster Systems*, Part no 3000230.1291

As stated earlier, how much CPU resource a user will consume is determined by the CPU required by each transaction type and, to a lesser degree, the frequency of each transaction type. Because the methods described above to estimate CPU resource

are prone to error, we have provided an alternate method, flexible but reasonably accurate, for sizing CPU requirements in generic Oracle RDBMS-based tools and applications, to be used with a complete and accurate description of the application and user activity.

Our method consists of several tasks:

- **First**, use your descriptions of the application and user activity to classify several transaction types and how often they are used.
- **Second**, compare your transaction types to the transactions defined in the **Typical CPU Workloads** table, which defines a broad range of CPU use for several transaction types. For each of your transactions, identify a few transactions in this table that compare reasonably close, and note the CPU use estimated for the table transaction. Extrapolate between two similar transaction types to estimate CPU use.
- **Third**, multiply the estimated CPU requirement for each transaction by how often each user issues that transaction. The result is each user's total sustained CPU requirement.
- **Fourth**, use the transaction type descriptions in the **Typical Memory Requirements** table, which includes a broad range of Memory uses for several transaction types, to estimate each user's memory requirements.
- **Fifth**, use the estimates from these tables and the transaction frequencies in the spreadsheets (as documented in Appendix D) to total the CPU and memory resources required by all Oracle RDBMS users expected on the system.

You have already completed the first step. Now complete the second step, as shown in the example below.

Example

Compare the example database transaction descriptions in Step 1 to the data in the sizing tables. This application will be developed using SQL*Forms 3.0 and will run against an Oracle7 database. Therefore, each database transaction will be equivalent to some functional work path or cycle within one or multiple forms. These examples assume that each database transaction will eventually be performed within its own form.

So, comparing the description of the New Order database transaction from Step 1 with the database transaction descriptions in the sizing tables, we see that the New Order form will fall somewhere between a moderately light-weight form and a moderate-weight form, but be much closer to the latter. Therefore, you might roughly estimate CPU and memory requirements to be **37 VUP-seconds per transaction and 1.4 MB per user**.

Comparing the description of the Payment database transaction to the descriptions in the sizing tables, a form that automates the Payment transaction might resemble the light-weight form or be even slightly less intensive. Therefore, estimate CPU and memory requirements to be **12 VUP-seconds and 0.75 MB/user**.

The Order Status database transaction could be automated in another light-weight form similar to the light-weight form in the sizing tables, one slightly more intensive than the Payment form. Estimate CPU and memory requirements for this form to be **15 VUP-seconds/transaction and 0.75 MB/user**.

From the description of the Stock Level transaction in Step 1, a moderate to moderately heavy form might be required to provide the functionality. Therefore, estimate CPU and memory requirements to be **55 VUP seconds/transaction and 1.70 MB/user**.

The Delivery transaction is also intensive and processes even more rows than the Stock transaction. A slightly more resource-intensive form may be required. Estimating the CPU and memory resources as slightly higher than those for the Stock transaction would result in resource requirements of **65 VUP-seconds/transaction and 1.80 MB/user**.

Finally, we should consider the intermittent reporting activity performed by the administrative staff. This activity may not be frequent; however, reporting is typically resource-intensive, and can cause resource bottlenecks.

Ad-hoc reporting will be done using SQL*Plus, and the scheduled reporting will be done using SQL*ReportWriter. The SQL*Plus ad-hoc reports are likely to be moderate-weight to moderately heavy-weight because the administrative staff is not trained to use the SQL Language optimally. Therefore, resource estimates of **125 VUP-seconds/report (transaction) and 2.0 MB/user** are reasonable. The SQL*ReportWriter reports will be written and maintained by the MIS staff. These reports will likely resemble the moderate-weight batch reports in the sizing tables. So, estimate CPU and memory resources at **170 VUP-seconds/report and 1.5 MB/user**.

The Sizing Tables

Typical CPU Workloads Table (for Oracle7)

| <u>Transaction Type</u> | <u>CPU Workload in Units x Seconds per Transaction</u> |
|---|--|
| <i>SQL*Forms V3.0</i> | |
| <u>Heavy, Complex Form</u> Query-intensive, several updates, 100s of rows qualified or sorted, datatype conversions, very complex joins, nested views, complex user exists, several block- and row-level triggers. | <u>130.</u> |
| <u>Moderately Heavy Form</u> Complex, 100s of rows sorted/qualified, several updates, multiple views, few joins, block- or row-level triggers, a few simple user exits. | <u>65.</u> |
| <u>Moderate Form</u> Some sorting, 10s of rows qualified, moderate update activity, selective indexes, few joins or views, few triggers, no user exits. | <u>40.</u> |
| <u>Moderately Light Form</u> Few rows qualified, small sorts, some update activity, few joins or views, highly selective indexes, few triggers or datatype conversions. | <u>20.</u> |
| <u>Light Form</u> Mostly query, few rows qualified, no sorts, one simple join, one view, few tables, few triggers, no user exits. | <u>15.</u> |
| <i>Pro*C</i> | |
| <u>Complex, READ-Only Transaction (TPC-C Stock)</u> Complex query, two-way join, DISTINCT operator, 200 rows selected and sorted. | <u>9.0</u> |
| <u>Heavy READ/WRITE Transaction (TPC-C Delivery)</u> 40 single-row selects, 30 single-row updates, 10 deletes. | <u>8.5</u> |
| <u>Moderate READ/WRITE Transaction (TPC-C Order)</u> 23 single row selects, 12 single-row inserts, 11 single-row updates. | <u>4.0</u> |
| <u>Simple update (READ/WRITE) Transaction (TPC-C Payment)</u> Three single-row updates one single-row select, one single-row insert. | <u>0.60</u> |
| <u>Simple READ-Only Transaction (TPC-C Status)</u> One single-row select, one select with nested select using MIN function, one ten-row select. | <u>0.50</u> |
| <u>Simple READ/WRITE Transaction (TPC-B Transaction)</u> Three single-row updates, one single-row insert, one single-row select. | <u>0.40</u> |

| <u>Transaction Type</u> | <u>CPU Workload in Units x Seconds per Transaction</u> |
|--|--|
| <i>SQL*Plus</i> | |
| <u>Heavy, Complex READ/WRITE Transaction</u> In batch, several multi-row queries, several nested views, many two- or three-way joins, 100s to 1000's of rows sorted, many non-selective indexes, 10s of updates or inserts. | <u>450</u> |
| <u>Moderately Heavy READ/WRITE Transaction</u> In batch, 100s of rows selected, many joins/views, poor indexes, moderately heavy update/insert activity. | <u>200</u> |
| <u>Moderate READ/WRITE Transaction</u> In batch, 10 - 100s rows qualified or sorted, datatype conversions, complex joins and nested views, selective indexes, several updates. | <u>95</u> |
| <u>Moderately Light READ/WRITE Transaction</u> Some sorting, 10s of rows qualified, moderate update activity, selective indexes, few joins or views. | <u>55</u> |
| <u>Light READ/WRITE Transaction</u> Ad-hoc query, few rows selected, highly selective indexes, few column datatype conversions, 2/3 single-row updates. | <u>15</u> |
| <u>Simple Read-Only Transaction</u> Query on EMP and DEPT join, three rows qualified, simple calculation on column data, very small tables. | <u>.85</u> |
| <i>SQL*ReportWriter</i> | |
| <u>Heavy, Complex Batch Report</u> 100s - 1000s rows, very large tables, large sorts, one full table scan, poor indexes, very complex/nested views, several two- or three-way joins. | <u>500</u> |
| <u>Moderately Heavy Batch Report</u> 100s of rows selected many two- or three-way joins or complex views, large sorts, poor indexes, column calculations. | <u>300</u> |
| <u>Moderate Interactive or Batch Report</u> 100s of rows, considerable sort activity, datatype conversions, less selective indexes, several complex joins and views. | <u>150</u> |
| <u>Moderately Light Interactive Report</u> 10 to 100s of rows sorted, several small tables, few large tables, nested views, complex joins, very selective indexes. | <u>90</u> |
| <u>Very Light, Simple Interactive Report</u> 10s of rows, some sorting, few small tables, two or three large tables, highly selective indexes, few joins and/or views. | <u>40</u> |

Typical Memory Requirements Table (for Oracle7)

| <u>Transaction Type</u> | <u>Estimated Memory Required (MB/User)</u> | <u>Transaction Type</u> | <u>Estimated Memory Required MB/User</u> |
|---|--|---|--|
| <i>SQL*Forms V3.0</i> | | <i>SQL*Plus</i> | |
| <u>Heavy, Complex Form</u> | <u>2.50</u> | <u>Heavy, Complex READ/WRITE Transaction</u> | <u>2.00</u> |
| Query-intensive, several updates, 100s of rows qualified or sorted, datatype conversions, very complex joins, nested views, complex user exists, several block- and row-level triggers. | | In batch, several multi-row queries, several nested views, many two- or three-way joins, 100s to 1000's of rows sorted, many non-selective indexes, 10s of updates/inserts. | |
| <u>Moderately Heavy Form</u> | <u>1.80</u> | <u>Moderately Heavy READ/WRITE Transaction</u> | <u>1.75</u> |
| Complex, 100s of rows sorted or qualified, several updates, multiple views, few joins, block- or row-level triggers, a few simple user exits. | | In batch, 100s of rows selected, many joins/views, poor indexes, moderately heavy update/insert activity. | |
| <u>Moderate Form</u> | <u>1.50</u> | <u>Moderate READ/WRITE Transaction</u> | <u>1.50</u> |
| Some sorting, 10s of rows qualified, moderate update activity, selective indexes, few joins or views, few triggers, no user exits. | | In batch, 10 - 100s rows qualified or sorted, datatype conversions, complex joins and nested views, selective indexes, several updates. | |
| <u>Moderately Light Form</u> | <u>1.25</u> | <u>Moderately Light READ/WRITE Transaction</u> | <u>1.25</u> |
| Few rows qualified, small sorts, some update activity, few joins or views, highly selective indexes, few triggers or datatype conversions. | | Some sorting, 10s of rows qualified, moderate update activity, selective indexes, few joins or views. | |
| <u>Light Form</u> | <u>.75</u> | <u>Light READ/WRITE Transaction</u> | <u>.75</u> |
| Mostly query, few rows qualified, no sorts, one simple join, one view, few tables, few triggers, no user exits. | | One ad-hoc query, few rows selected, highly selective indexes, few column datatype conversions, 2/3 single-row updates. | |
| <i>Pro*C</i> | | <u>Simple Read-Only Transaction</u> | <u>.50</u> |
| <u>Complex, READ-Only Transaction (TPC-C Stock)</u> | <u>2.00</u> | Query on EMP and DEPT join, three rows qualified, simple calculation on column data, very small tables. | |
| Complex query, two-way join, DISTINCT operator, 200 rows selected and sorted. | | <i>SQL*ReportWriter</i> | |
| <u>Heavy READ/WRITE Transaction (TPC-C Delivery)</u> | <u>1.75</u> | <u>Heavy, Complex Batch Report</u> | <u>2.20</u> |
| 40 single-row selects, 30 single-row updates, 10 deletes. | | 100s - 1000s rows, very large tables, large sorts, one full table scan, poor indexes, very complex/nested views, several 2/3-way joins. | |
| <u>Moderate READ/WRITE Transaction (TPC-C Order)</u> | <u>1.50</u> | <u>Moderately Heavy Batch Report</u> | <u>1.85</u> |
| 23 single row selects, 12 single-row inserts, 11 single-row updates. | | 100s of rows selected many two- or three-way joins and complex views, large sorts, poor indexes, column calculations. | |
| <u>Simple Update (READ/WRITE) Transaction (TPC-C Payment)</u> | <u>1.25</u> | <u>Moderate Interactive or Batch Report</u> | <u>1.50</u> |
| Three single-row updates one single-row select, one single-row insert. | | 100s of rows, considerable sort activity, datatype conversions, less selective indexes, several complex joins and views. | |
| <u>Simple READ-Only Transaction (TPC-C Status)</u> | <u>0.75</u> | <u>Moderately Light Interactive Report</u> | <u>1.20</u> |
| One single-row select, one select with nested select using MIN function, one ten-row select. | | 10 to 100s of rows sorted, several small tables, few large tables, nested views, complex joins, very selective indexes. | |
| <u>Simple READ/WRITE Transaction (TPC-B Transaction)</u> | <u>0.75</u> | <u>Very Light, Simple Interactive Report</u> | <u>.80</u> |
| Three single-row updates, one single-row insert, one single-row select. | | 10s of rows, some sorting, few small tables, two or three large tables, highly selective indexes, few joins and/or views. | |

*** Step 3. Predict Total CPU and Memory Resource Use**

Step 3 consists of the following tasks:

- Total your CPU and memory use estimates.
- Estimate total I/O load (rate).
- Determine the optimum number of disk drives.
- Consider other system resource requirements.

Step 3a. Total CPU and Memory Use Requirements

When sizing:

Now that you've estimated the CPU and memory requirements for each transaction type, determine the total CPU and memory required by the entire production system using the following formulas:

$$V_t = \text{SUM}(i=1,n)[N_i * R_i * \text{CPU}_i]$$

$$M_t = \text{SUM}(i=1,n)[N_i * M_i]$$

Where: is:

V_t is the value to be used with the customer's performance expectations to determine the minimum uniprocessor CPU rating for the future system.

M_t the total memory that all Oracle RDBMS-based users will consume, not including the SGA.

CPU_i the estimated CPU workload for the i th transaction type. (Units are "CPU-units * seconds" per transaction.)

R_i the expected txn frequency/user for the i th transaction. (Units are transactions/second/user.)

N_i the number of users issuing the i th transaction. (Units are number of users.)

M_i the estimated memory for one user of the i th transaction. (Units are MB per user.)

n the total number of transaction types considered.

Sizing Worksheet

| Transaction type i | N_i users | R_i rate | CPU_i | M_i | CPU | Mem. |
|----------------------|-------------|------------|----------------|-------|-------|-------|
| Txn 1 | a | b | c | d | abc | ad |
| Txn 2 | e | f | g | h | efg | eh |
| Txn 3 | j | l | p | q | jlp | jq |
| Txn n | w | x | y | z | wxy | wz |
| Totals: | | | | | V_t | M_t |

Example

From the description obtained in Step 1, we know the following information about the transaction frequencies:

- 30 interactive users:
 - Order forms (3500 per 8-hours period)
 - Payment forms (3500 per 8-hour period)
 - Status forms (500 per 8-hour period)

This works out to be roughly 14 interactive users processing New Order requests, 14 users processing Payments, and two users processing Status requests. In production, each interactive user would process a mixture of these transactions.

Four of the warehouse managers will handle the Delivery processing and one will handle the Stock Level requests:

- 5 warehouse managers:
 - Stock forms (two per 1-hour period)
 - Delivery forms (7500 per-16 hours)
- 2 Administrative staff:
 - SQL*Plus reports (one per 1-hour period)
 - SQL*ReportWriter (one per 1-hour period)

Units are transactions/second/user for rates in the table below.

Sizing Worksheet

| Trans. type i | N_i users | R_i rate | CPU_i | M_i | CPU | Mem. |
|-----------------|-------------|------------|----------------|-------|-------|------|
| Txn 1 | 14 | .0087 | 37 | 1.4 | 4.51 | 19.6 |
| Txn 2 | 14 | .0087 | 12 | 0.75 | 1.46 | 10.5 |
| Txn 3 | 2 | .0087 | 15 | 0.75 | 0.26 | 1.5 |
| Txn 4 | 1 | .00056 | 55 | 1.7 | 0.031 | 1.7 |
| Txn 5 | 4 | .0326 | 65 | 1.8 | 8.48 | 7.2 |
| Txn 6 | 1 | .00028 | 125 | 1.75 | 0.035 | 2.0 |
| Txn 7 | 1 | .00028 | 170 | 1.50 | 0.048 | 1.5 |
| Totals: | | | | | 14.82 | 44.0 |

When planning capacity:

Now that you've measured the CPU and memory requirements for each transaction type, determine the total CPU and memory required by the entire production system using the following formulas:

$$V_t = \text{SUM}(i=1,n)[N_i * R_i * \text{CPU}_i]$$

$$M_t = \text{SUM}(i=1,n)[N_i * M_i]$$

Where: is:

- M_t the total memory that all Oracle RDBMS based users will consume, not including the SGA.
- V_t is the value to be used with the customer's performance expectations, to determine the minimum uniprocessor CPU rating for the future system.
- CPU_i the normalized CPU workload measured for the i th transaction type. (Units are "CPU-units * seconds" per transaction.)
- R_i the expected txn frequency/user for the i th transaction.(Units are transactions/second/user.)
- N_i the number of users issuing the i th transaction. (Units are number of users.)
- M_i measured memory for one user of the i th transaction. (Units are MB per user.)
- n the total number of transaction types considered.

Capacity Planning Worksheet

| Transaction type i | N_i users | R_i rate | CPU_i | M_i | CPU | Mem. |
|----------------------|-------------|------------|----------------|-------|-------|-------|
| Txn 1 | a | b | c | d | abc | ad |
| Txn 2 | e | f | g | h | efg | eh |
| Txn 3 | j | l | p | q | jlp | jq |
| Txn n | w | x | y | z | wxy | wz |
| <i>Totals:</i> | | | | | V_t | M_t |

Step 3b. Estimate Total I/O Load

You can also estimate the I/O load (rate) that each user type will place on the system. Then you can use the I/O load to determine the optimal number of disk drives for the system.

When sizing:

It is extremely difficult, if not impossible, to predict accurately the total system I/O rate due to the Oracle Server-related activity. Without completely understanding both the effectiveness of the SGA size (including the process private SORT area sizes) and the physical structure of the database (including size, layout, indexes, cardinality, etc.), you cannot accurately predict the amount of I/O that will be generated by any transaction type.

Rather than collect information on all of the factors which will affect the total I/O rate, you can follow the quick and simple method of grossly estimating the total I/O rate presented here.

Testing has shown that a linear correlation exists between the amount of logical I/O (LIO) performed by a particular transaction and the amount of CPU used to process that transaction. (The slope of this correlation for VAX OpenVMS and Oracle7 varies between **.007** and **.011 VUP-seconds/LIO**). The CPU time, however, is only that spent performing actual database activity, not the CPU time used by any end-user tool.

The general method for sizing has produced a way to estimate the total amount of CPU needed (V_t) to support an application and its expected workload, but it does not indicate what percentage of this CPU load is dedicated to the database activity or to end-user tools. So, to estimate the logical I/O that an application workload will perform, the following estimate considers all projected CPU requirements to be the result of database activity only:

High estimate:

$$\text{LIO/SEC}_{hi} = V_t / (.007 \text{ VUP-seconds/LIO})$$

Low estimate:

$$\text{LIO/SEC}_{lo} = V_t / (.011 \text{ VUP-seconds/LIO})$$

Having determined this gross range of estimated logical I/O (LIO) performed by the expected application/workload, you can now use an estimated cache hit ratio (CHR) value to estimate the total physical I/O (PIO) which this application/workload will generate.

Note: This is a very rough estimation method; do not rely on it as an absolutely accurate calculation of physical I/O requirements.

High estimate:

$$\text{PIO/SEC}_{hi} = (1 - \text{CHR}) * \text{LIO/SEC}_{hi}$$

Low estimate:

$$\text{PIO/SEC}_{lo} = (1 - \text{CHR}) * \text{LIO/SEC}_{lo}$$

Example

Using the equation above, with various values for the cache hit ratio, you can determine several estimates for the range of logical I/O to be generated:

| | | |
|-----------------------|------------------------------------|-------------|
| | $V_t = 14.82$ | $CHR = .80$ |
| LIO/SEC _{hi} | $14.82/0.007 = 2117.1 = \sim 2100$ | LIO/SEC |
| LIO/SEC _{lo} | $14.82/0.011 = 1347.3 = \sim 1350$ | LIO/SEC |
| PIO/SEC _{hi} | $(1 - .80) * (2100) = 420$ | PIO/SEC |
| PIO/SEC _{lo} | $(1 - .80) * (1350) = 270$ | PIO/SEC |
| | | |
| | $V_t = 14.82$ | $CHR = .85$ |
| PIO/SEC _{hi} | $(1 - .85) * (2100) = 315$ | PIO/SEC |
| PIO/SEC _{lo} | $(1 - .85) * (1350) = 203$ | PIO/SEC |
| | | |
| | $V_t = 14.82$ | $CHR = .90$ |
| PIO/SEC _{hi} | $(1 - .90) * (2100) = 210$ | PIO/SEC |
| PIO/SEC _{lo} | $(1 - .90) * (1350) = 135$ | PIO/SEC |
| | | |
| | $V_t = 14.82$ | $CHR = .95$ |
| PIO/SEC _{hi} | $(1 - .95) * (2100) = 105$ | PIO/SEC |
| PIO/SEC _{lo} | $(1 - .95) * (1350) = 68$ | PIO/SEC |

When planning capacity:

The method for determining the total expected physical I/O rate when planning capacity is considerably more scientific than the method used when sizing. The total physical I/O rate for an Oracle RDBMS system is:

$$\text{PIO/sec} = \text{IO/sec}_{(\text{other})} + \text{SUM}(i=1,N)[\text{IO}_i/\text{txn} * R_i]$$

Where: is:

| | |
|---------------------------|--|
| IO/sec _(other) | the I/O consumed by processes other than the applications |
| IO _i | the number of I/O incurred by the <i>i</i> th transaction type |
| R _i | the expected frequency or arrival rate of the <i>i</i> th transaction. |

(Note that I/O charged to a transaction is usually READ I/O because the Oracle Server background processes manage all of the database WRITE I/O activity.)

When planning capacity, you can determine the average number of I/O (IO_i) of an average transaction by using the BSTAT/ESTAT or TKPROF utilities on a system running under a typical workload.

Having collected this data, complete the following tasks:

- Multiply the measured IO_i for each transaction type by the expected peak frequency (R_i) of that transaction.
- Then, total the results for each transaction type.
- Account for any other I/O load due to the Oracle Server instance background processes or not related to this application.

Normally, when estimating the write I/O rate, you will need to consider only the DBWR and the LGWR processes.

DBWR performs all physical writes to the database files, except for the redo log files. So, the number of physical I/Os for DBWR is the total number of physical writes for the instance.

LGWR writes a redo log to the redo log file each time a database transaction is committed. LGWR may write the redo logs for more than one transaction at the same time. So, you can estimate the number of physical writes for LGWR as the number of database transactions executed in the same time frame multiplied by 0.4 - 1.0. If you use a dedicated disk for the redo log files, do not include disk I/O by LGWR, and exclude the disk when calculating the capacity of the disk subsystem.

Step 3c. Determine Optimum Number of Disk Drives

Assume the total system-wide physical I/O rate (PIO/sec) due to Oracle RDBMS users will be evenly balanced across all the disk drives available to the Oracle RDBMS. Normally, a disk drive can handle 20-30 I/Os per second. (Refer to the vendor documentation to find out the exact number for your disks.) Optimum performance is probably achieved at a fraction of this total I/O throughput capacity. We will use 25% to approximate.

By assuming the total physical I/O rate will be balanced across all the disk drives, we can estimate the required number of disks (D) for optimum performance as:

$$D = \frac{\text{total PIO/sec}}{(\text{the I/O throughput capacity of a disk} * 25\%)}$$

where PIO/sec is the total number of physical disk I/Os per second generated by this application in the peak workload period.

Now, calculate the total I/O throughput capacity (T) across all "D" disks as:

$$T = D * \text{the I/O throughput capacity of a disk}$$

The average disk throughput utilization (U_d) is:

$$U_d = \frac{\text{total PIO/sec}}{T} = .25 = 25\%$$

The disk utilization U_d is very important because it will directly affect the actual response time for each disk I/O which in turn will affect the user's interactive response time. **For overall system loading, the number of disk drives is more significant than their total storage capacity.** This is to allow for more efficient balancing of disk I/O activity. Ideally, the redo log files should always reside on a dedicated disk(s) to improve the performance of the sequential I/O to the open redo log file.

Example

Having several ranges for estimated physical I/O rates, you can estimate how many disks will ensure good performance. Use the equation given in this section, with 40 I/Os per second as the I/O throughput capacity for a disk drive.

Note: There is no consideration here for storage capacity.

CHR = .80
 $PIO/SEC_{hi} \quad (1 - .80) * (2100) = 420 \text{ PIO/SEC}$
 $PIO/SEC_{lo} \quad (1 - .80) * (1350) = 270 \text{ PIO/SEC}$

“D” number of disks:
 $420/(40*.25) = 42 \quad \text{to} \quad 270/(40*.25) = 27$

CHR = .85
 $PIO/SEC_{hi} \quad (1 - .85) * (2100) = 315 \text{ PIO/SEC}$
 $PIO/SEC_{lo} \quad (1 - .85) * (1350) = 203 \text{ PIO/SEC}$

“D” number of disks:
 $315/(40*.25) = 32 \quad \text{to} \quad 203/(40*.25) = 20$

CHR = .90
 $PIO/SEC_{hi} \quad (1 - .90) * (2100) = 210 \text{ PIO/SEC}$
 $PIO/SEC_{lo} \quad (1 - .90) * (1350) = 135 \text{ PIO/SEC}$

“D” number of disks:
 $210/(40*.25) = 21 \quad \text{to} \quad 135/(40*.25) = 14$

CHR = .95
 $PIO/SEC_{hi} \quad (1 - .95) * (2100) = 105 \text{ PIO/SEC}$
 $PIO/SEC_{lo} \quad (1 - .95) * (1350) = 68 \text{ PIO/SEC}$

“D” number of disks:
 $105/(40*.25) = 11 \quad \text{to} \quad 68/(40*.25) = 7$

Note that the optimum number of disks depends on the value of the cache hit ratio, underlining the need for properly tuning the overall system. Assuming the cache hit ratio is greater than or equal to .90 (because the system is expected to be well-tuned), from 14 to 21 disks are needed for this application workload. For convenience, we'll choose **15** disks for this example.

Because most disk drives can support a maximum I/O throughput of about 40 I/Os per second, the total I/O rate throughput capacity (T) of all 15 disks combined will be 600 I/Os per second. Define the value of PIO/SEC_{hi} as 210 I/Os per second. From the chosen range of CHR=.90, we can estimate the peak average utilization of these disks, assuming I/O is balanced across all disks which WILL affect the user's response time as described in Step 4:

$$U_d = 210/600 = .35 \text{ or } 35\%$$

Considering disk space requirements of Oracle RDBMS software

Because we cannot forecast the overall database size without extensive information about the nature, structure, amount, etc. of the data, these disk space calculations do not include the size of the actual database files, redo log files, and control files. Only the disk space required by the Oracle RDBMS software is included.

On the next page, you will find a list of approximate values for the disk space required for Oracle7 products, by directory. These requirements include space for such product sub-directories as DEMO.

The values are in OpenVMS disk blocks, where one disk block equals 512 bytes. The values are approximate, and will vary according to how the products are linked, how often the directories are purged, etc. Space requirements between incremental versions of the products may also vary. For information about the space required to install products as shared images, see the *Oracle7 for DEC VAX VMS Installation and User's Guide*.

| <u>Directory</u> | <u>VMS Blocks</u> | <u>Comments</u> |
|----------------------|-------------------|--|
| CGSPKG | 6100 | Graphics routines used by Easy*SQL and SQL*Graph |
| CRT | 2100 | Required terminal definitions file |
| EASYSQL | 7300 | |
| INSTALL | 1700 | Required installation scripts |
| NETCONFIG | 2400 | SQL*NET V1.2 common code and VMS Mailbox driver |
| NLS | 11000 | Multilingual option |
| OPTION | 100 | Switch for transaction processing option (PL/SQL code has its own directory) |
| ORACLEMAIL | 29500 | |
| ORACLETERM- INAL | 4000 | |
| PLSQL | 2600 | PL/SQL component of TPO option |
| PROGINT | 12400 | Pro*C, Pro*Ada, Pro*PL/I, Pro*COBOL, Pro*FORTRAN, Pro*Pascal |
| RDBMS | 2100 | |
| SQLCALC | 2800 | |
| SQLFORMS | 7700 | SQL*Forms V2.3 |
| SQL FORMS30 | | SQL*Forms V3.0 |
| SQLGRAPH | 9100 | |
| SQLLOADER | 1300 | |
| SQLMENU | 4100 | SQL*Menu V4.1 |
| SQLMENU50 | 13300 | SQL*Menu V5.0 |
| SQLNETASY | 230 | SQL*Net Async driver V1.2 |
| SQLNETDNT | 70 | SQL*Net DECnet driver V1.2 |
| SQLNETTCP | 1000 | SQL*Net TCP/IP driver V1.2 |
| SQLNET | 1900 | SQL*Net V1.1 (includes all V1.1 drivers) |
| SQLPLUS | 6000 | |
| SQLQMX | 3600 | |
| SQLREPORT | 1600 | |
| SQLREPORT- WRITER | 19400 | |
| SQLTR | 20000 | SQL*TextRetrieval |
| UTIL | 600 | Required installation utilities |
| Total | 211900 | |
| CASE*Designer | 28459 | |
| CASE*Dictionary | 66664 | |
| CASE*Generator | 24 | |
| Total | 119839 | |

Step 3d.

Consider Other System Resource Requirements

Before completing Step 3, you should consider any resource requirements which will be placed upon the system, other than those due explicitly to the Oracle RDBMS-based processes:

Other CPU requirements:

The actual CPU load (V_t) should include any other CPU load, including the CPU load caused by the background Oracle Server processes:

$$\text{CPU Load} = \text{CPU}_{(\text{other})} + \text{SUM}(i=1,n)[N_i * \text{CPU}_i * R_i]$$

Where: is:

n the number of transaction types.

$\text{CPU}_{(\text{other})}$ the CPU load incurred by the processes other than the applications to be sized.

N_i the number of users issuing the i th transactions.

CPU_i the CPU portion of the Transaction Workload Profile for the i th transaction.

R_i the expected frequency or arrival rate of the i th type transaction.

Other memory requirements

The actual amount of memory required (M_t) for an Oracle RDBMS application should be expressed as:

$$M_t = M_{\text{sys}} + M_{\text{ker}} + M_{\text{sga}} + M_{\text{bk}} + \text{SUM}(i=1,n)[M_i * N_i] + M_{\text{other}}$$

Where: is:
 M_{sys} the memory used by VMS. Refer to your Digital documentation to determine this value. Certain SYSGEN parameters will affect how much memory VMS uses: these parameters include the size of the global page table, the IRP/SRP/LRP counts, the system working set, etc.

M_{ker} memory space for the Oracle Server kernel image, which is shared by both the user processes and the Oracle Server background processes. The memory used by the kernel image M_{ker} is approximately 1.0 - 1.2 MB. If more than one Oracle RDBMS instance runs and they use different Oracle Server image files, use the combined sizes of these images for M_{ker} .

M_sga the memory space used by the Oracle SGA, which is shared by all the processes running on an Oracle Server instance. If more than one instance runs in the time frame, total the memory used by each instance's SGA. The optimum SGA size differs for different applications. M_bk the private space used by Oracle Server background processes. For an active instance, the amount of private memory space used by the background processes is approximately:

DBWR 0.8 MB
 LGWR 0.65 MB
 SMON 0.65 MB
 PMON 0.65 MB

If there are more than one Oracle Server instance, total the background processes for each instance.

M_i the amount of private memory used by a user process that issues only the *i*th transaction type.

N_i the number of users that issue the *i*th transaction type.

M_other the memory used by any non-Oracle RDBMS users.

*** Step 4. Model CPU capacity Requirements**

Before you begin Step 4, define what your customer considers acceptable performance for each transaction type. You should define acceptable performance in terms of what is acceptable total response time. "Total response time" is the amount of elapsed time taken to complete all transaction components minus "think" time and typing time of course.

Total CPU time consumed executing a transaction (service time)
 + Time spent waiting for the CPU to be free (wait time)
 + I/O transfer time + I/O wait time
 = Total Response Time

Note: Together, CPU service time and CPU wait time comprise CPU processing time. Together, I/O transfer time and I/O wait time comprise I/O processing time.

Step 4 consists of three tasks:

- Use your total estimated CPU requirement (V_t) to identify potential CPU models.
- Then, apply the queuing model described in this section using the values determined for:
 CPU_i for each transaction type
 acceptable total response time for each transaction type
 V_t determined in Step 3

- to find the minimum uniprocessor CPU rating needed.
- Choose a system and use its uniprocessor CPU rating (V_c) to estimate the impact on processing and response times (You can find V_c values in Appendix C.)

Step 4a.

Identify Potential CPU Models

Now that you've defined the total CPU capacity required to support the expected workload, identify which CPU models will meet these requirements. Choose potential CPU models according to the expected use as well as total CPU requirements. For example, if you estimated total CPU use to be between 10 and 14 VUPs, any of the following VAX models might suffice, although perhaps not most efficiently:

| <u>Model</u> | <u>VUP Rating</u> |
|---------------------------------------|--------------------------------|
| VAX 4000 model 500 | 24 VUPs |
| VAX 6000 model 360 | 22 VUPs |
| VAX 6000 model 430 or above | 19, 25, 31, 36 VUPs |
| VAX 6000 model 520 or above | 25, 37, 49, 61, 72 VUPs |
| VAX 6000 model 610 or above | 32, 58, 84, 106, 128, 150 VUPs |
| VAX 7000 model 610 or above | 36 VUPs |
| Any VAX 9000 (possibly an older 8840) | 40 VUPs and higher (22 VUPs) |
| VAX 10000 model 610 or above | 36 VUPs |

The CPU rating of a CPU model may be sufficient to handle the expected CPU load (V_t), but that system may still not meet your customer's performance expectations. A system that is 80 - 100% used will likely result in poor response times for end-users. Therefore, when sizing consider the end-users' performance expectations as well, and select a CPU that will handle the expected CPU load with a significant margin of idle time.

Example

Several CPU models have aggregate CPU ratings larger than the V_t of 14.82 (~15) estimated in Step 3. The following list was taken from Appendix C:

| | |
|---------------------------|---------------------|
| VAX 8840 | 21 VUPs (4 CPUs) |
| VAX 3100 model 90 | 24 VUPs |
| VAX 4000 model 90 | 24 VUPs |
| VAX 4000 model 100 | 24 VUPs |
| VAX 4000 model 500 | 24 VUPs |
| VAX 4000 model 600 | 32 VUPs |
| VAX 6000 model 360 | ~17.8 VUPs (6 CPUs) |
| VAX 6000 model 430 and up | ~ 19 VUPs (3 CPUs) |
| VAX 6000 model 520 and up | ~ 25 VUPs (2 CPUs) |
| VAX 6000 model 610 and up | 32 VUPs (1 CPU) |
| VAX 7000 model 610 and up | ~ 36 VUPs (1 CPU) |
| VAX 9000 model 210 and up | 40 VUPs (1 CPU) |

Step 4b.

Calculate Minimum Uniprocessor CPU Rating

The following formula is derived from the steady state formula for an M/M/c queuing model. Use it to determine the minimum uniprocessor CPU rating that will meet your customer's performance expectations:

$$V = \frac{\text{SUM}_{i=1,n} [N_i * CPU_i]}{\text{SUM}_{i=1,n} [N_i * Z_i]} + \frac{V_t}{K}$$

(See appendix B for this formula derivation.)

How to use the queuing model:

First, recall the expected number of users of each type of transaction determined in Step 1 of the process using the customer survey.

N_i is the number of users of the i th transaction type.

Second, recall the CPU workload that you either measured or obtained by comparison to the Typical CPU Workloads table.

CPU_i is the CPU workload or normalized CPU time (in CPU-units * seconds) required for the i th transaction type.

Third, for each transaction type, ask your customer to define *acceptable performance* on the new system (RT_i), otherwise known as the "acceptable total response time." As stated earlier, the term "total response time" refers to the total amount of elapsed time taken to complete all the pieces of a transaction, minus "think" time and typing time.

RT_i is the acceptable total response time for the i th transaction type.

Again, the two major components of total response time (RT_i) are:

- Z_i , CPU execution and wait time, also referred to as the "CPU processing time"
- IOT_i , I/O transfer time and wait time, also referred to as the "I/O processing time"

These components are used in different areas of the queuing model to convert total response time, RT_i , into its two components:

$$RT_i = Z_i + IOT_i$$

How to find I/O processing time (IOT_i) for each transaction

Either estimate or measure the number of physical I/Os the transaction performs (IO_i). Then, multiply this number by the average time needed to perform an Oracle RDBMS physical I/O (approximately **.025 seconds** for most Digital

systems) to obtain the total I/O transfer time (tt_i). Next, allow for the I/O wait time by using the I/O transfer time in the steady state formula for an M/M/1 queuing model.

$$tt_i = IO_i * (.025 \text{ sec. per I/O})$$

$$IOT_i = \frac{tt_i}{(1 - U_d)}$$

$$IOT_i = \frac{IO_i * (.025 \text{ seconds per I/O})}{(1 - U_d)}$$

When sizing, you should already know the CPU requirement for the i th transaction in CPU-units * seconds (CPU_i). So, you can roughly estimate IO_i using your estimated cache hit ratio (CHR) and the loose correlation between CPU and logical I/O (LIO) mentioned earlier. This calculation is not an absolutely accurate prediction of the I/O generated by a transaction, but it effectively reduces your customer's response time performance criteria into the components required by the queuing theory formula.

$$IO_i = (CPU_i/0.009)*(1-CHR)$$

[use 0.009 as the correlation factor for *average* I/O rate]

Therefore:

$$IOT_i = \frac{(CPU_i/0.009)*(1-CHR)* (0.025 \text{ seconds per I/O})}{(1 - U_d)}$$

How to find CPU processing time (Z_i) for each transaction

You should have already determined total acceptable response time, or RT_i , from the performance criteria collected through the customer survey. The value of Z_i is the value of RT_i minus the value of IOT_i determined above:

$$Z_i = RT_i - IOT_i$$

Z_i represents the worst-case CPU processing time acceptable to your customer for each transaction type.

Now, apply the steady state formula for an M/M/c queuing model to derive the minimum uniprocessor CPU rating (V) that will support both the total expected workload (V_t) and the acceptable "CPU processing times" (Z_i) for each transaction type. The only variable in the formula is the value of SMP Factor K. Obtain values for SMP Factor K from the table on the next page. Using the K value for each type of SMP system (i.e., two or more CPUs), calculate several values for the minimum uniprocessor CPU rating.

$$V = \frac{\sum_{i=1,n} [N_i * CPU_i]}{\sum_{i=1,n} [N_i * Z_i]} + \frac{V_t}{K}$$

(See appendix B for this formula derivation.)

Choose a system with uniprocessor CPU rating (V_c) that meets or exceeds this recommended minimum uniprocessor CPU rating (V). Refer to the table in Appendix C.

SMP Factor K Table

| <u>Model</u> | <u>"c" CPUs/Kcpu = K</u> | |
|--------------|--------------------------|--------|
| 6610 | 1/1 | = 1.00 |
| 6620 | 2/1.25 | = 1.60 |
| 6630 | 3/1.29 | = 2.33 |
| 6640 | 4/1.31 | = 3.05 |
| 6650 | 5/1.35 | = 3.70 |
| 6660 | 6/1.35 | = 4.44 |

Use the K values from this table in the queuing model formula. (Kcpu reflects the overhead caused by cooperation among the processors.) These factors represent approximately 70% scalability.

Example

You know N_i and CPU_i from Step 3. Now, include the performance expectations of the end-users, in the form of response times (RT_i) to estimate CPU requirements more accurately. You obtained values for RT_i from the survey completed in Step 1:

| | |
|-------------------------------|----------------------------|
| New Order transaction/form | 20 seconds |
| Payment transaction/form | 20 seconds |
| Order Status transaction/form | 20 seconds |
| Stock level transaction/form | 30 seconds |
| Delivery transaction/form | 90 seconds |
| Admin. Reports | 5 minutes (300 seconds) |

Convert response times (RT_i) values into allowable CPU processing time (Z_i) values by subtracting I/O transfer times (IOT_i), using several values for the cache hit ratio.

First, estimate the IOT_i for each transaction using the correlation, the cache hit ratio, and the M/M/1 steady state formula using $U_d = 35\%$ from Step 3c:

| | CHR | | | |
|-----------|--|-------------------|-------------------|-------------------|
| | <u>.80</u> | <u>.85</u> | <u>.90</u> | <u>.95</u> |
| New order | $IOT = (37/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>31.6</u> <u>23.7</u> <u>15.8</u> <u>7.9</u> | | | |
| Payment | $IOT = (12/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>10.3</u> <u>7.7</u> <u>5.1</u> <u>2.6</u> | | | |
| Status | $IOT = (15/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>12.8</u> <u>9.6</u> <u>6.4</u> <u>3.2</u> | | | |
| Stock | $IOT = (55/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>47.0</u> <u>35.3</u> <u>23.5</u> <u>11.8</u> | | | |
| Delivery | $IOT = (65/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>55.6</u> <u>41.7</u> <u>27.8</u> <u>13.9</u> | | | |
| SQL*Plus | $IOT = (125/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>106.8</u> <u>80.1</u> <u>53.4</u> <u>26.7</u> | | | |
| SQL*RW | $IOT = (170/0.009) * (1-CHR) * (0.025) / (1-.35)$ <u>145.3</u> <u>109.0</u> <u>72.7</u> <u>36.3</u> | | | |

In a well-tuned environment, the cache hit ration should be near or above .90, so we'll use this value to make the following calculations. (For several transaction types, smaller cache hit ratios may result in IOT values too large to meet even the end-users' definition of acceptable response times.)

Now subtract IOT_i from RT_i to obtain Z_i , the maximum allowable CPU processing time (execution and wait time) for the i th transaction (measured in seconds):

| | | | | | |
|-----|--------|---|---------|---|-------|
| | RT_i | - | IOT_i | = | Z_i |
| Z1= | 20 | - | 15.8 | = | 4.2 |
| Z2= | 20 | - | 5.1 | = | 14.9 |
| Z3= | 20 | - | 6.4 | = | 13.6 |
| Z4= | 30 | - | 23.5 | = | 6.5 |
| Z5= | 90 | - | 27.8 | = | 62.2 |
| Z6= | 300 | - | 53.4 | = | 246.6 |
| Z7= | 300 | - | 72.7 | = | 227.3 |

The values for N_i , CPU_i , and Z_i are now known or estimated and shown below.

| <u>Transaction</u> | <u>N_i</u> | <u>CPU_i</u> | <u>Z_i</u> |
|-------------------------------|-----------------------------|-------------------------------|-----------------------------|
| New Order transaction/form | 14 | 37 | 4.2 |
| Payment transaction/form | 14 | 12 | 14.9 |
| Order Status transaction/form | 2 | 15 | 13.6 |
| Stock level transaction/form | 1 | 55 | 6.5 |
| Delivery transaction/form | 4 | 65 | 62.2 |
| SQL*Plus ad-hoc report | 1 | 125 | 246.6 |
| SQL*ReportWriter batch report | 1 | 170 | 227.3 |

Choose several values for K (representing the number of CPUs and scalability in an SMP system) and for each value, use the data collected with the queuing model formula to determine the minimum uniprocessor CPU rating "V" (not the aggregate CPU rating) relative to the number of CPUs.

The Queuing Model Formula

$$V = \frac{\text{SUM } i=1,n[N_i * \text{CPU}_i]}{\text{SUM } i=1,n[N_i * Z_i]} + \frac{V_t}{K}$$

Minimum Uniprocessor CPU Rating

| | | | |
|-------------------|-----|-------|-----------|
| For 1 CPU K=1 | V = | ~16.1 | (17 VUPs) |
| For 2 CPUs K=1.60 | V = | ~10.6 | (11 VUPs) |
| For 3 CPUs K=2.33 | V = | ~7.7 | (8 VUPs) |
| For 4 CPUs K=3.05 | V = | ~6.2 | (7 VUPs) |
| For 5 CPUs K=3.70 | V = | ~5.3 | (6 VUPs) |
| For 6 CPUs K=4.44 | V = | ~4.6 | (5 VUPs) |

Based upon these estimations, a VAX 6000 model 520 (13 VUP uniprocessor rating with two CPUs) would handle this workload and meet the stated performance requirements.

An alternative might be a VAX 6000 model 430 (7 VUP uniprocessor with three CPUs) but this system may be slightly undersized if the estimating process was accurate. A VAX 6000 model 610 should also handle the workload but would probably be oversized.

Now choose a system. The uniprocessor rating of this system will be the value **V_c**.

Step 4c. Estimate the Impact of V_c and D on Total Response Times

Use the SMP Factor for the number of CPUs on the chosen system, the total CPU workload (V_t), the minimum uniprocessor CPU rating (V), the uniprocessor CPU rating of the chosen system (V_c), the estimated disk utilization from Step 3c (U_d), and the estimated cache hit ration (CHR) in the formula below to estimate the impact (X) of the chosen system and number of disks on acceptable response times.

$$X = \frac{\frac{1}{V_c - (V_t/K)} + \frac{(1/0.009) * (1 - CHR) * (0.025)}{(1 - U_d)}}{\frac{1}{V - (V_t/K)} + \frac{(1/0.009) * (1 - CHR) * (0.025)}{(1 - .25)}}$$

See Appendix B for this formula derivation.

On the chosen hardware configuration, the actual total response times for the expected peak CPU and disk workload may be up to **X** times the acceptable total response times used in the queuing model.

Example

Using a VAX 6520 and 15 disks:

$$X = \frac{\frac{1}{13 - (14.82/1.6)} + \frac{(1/0.009) * (1 - .90) * (0.025)}{(1 - .35)}}{\frac{1}{10.6 - (14.82/1.6)} + \frac{(1/0.009) * (1 - .90) * (0.025)}{(1 - .25)}}$$

In this example, X equals 0.625, so the actual response times may be up to 0.625 times the stated response time requirements. This is obviously acceptable.

Decreasing the number of disks chosen in Step 3 from 15 to 10 raises the average disk use from 35% to 50%. Recalculating the example equation using .50 for U_d, X is equal to 0.85, which is still acceptable but may induce I/O bottlenecks.

By decreasing the cache hit ratio (allowing for poor tuning, inefficient SGA, etc.) from .90 to .87, the original value for X becomes 2.18. This very clearly illustrates the very significant effect of disk utilization and/or cache hit ratio on overall response time. **So, configure the production system with sufficient disk drives to avoid disk bottlenecks, and tune the environment well.**

Example

Using a VAX 6000 model 430 and 15 disks:

$$X = \frac{\frac{1}{7 - (14.82/2.33)} + \frac{(1/0.009) * (1 - .90) * (0.025)}{(1 - .35)}}{\frac{1}{7.2 - (14.82/2.33)} + \frac{(1/0.009) * (1 - .90) * (0.025)}{(1 - .25)}}$$

X equals 1.16, a marginally unacceptable value. Adding additional disks reduces the value of X to 1.07, but the CPU utilization is already over 80% so there is idle time to insure acceptable interactive response times for the users. Raising the cache hit ratio, through tuning, will improve X (to under 1.00) but will not offload the over-utilized CPU.

**Appendix A
Customer Survey: What to Ask Your Customer**

Step 1. Define Your Customer's Expectations

The first and most important step in sizing is to quantify your customer's workload and performance expectations. How much detail you obtain about these expectations determines how accurate your sizing will be.

"How Do You Measure Performance?"

- Maximum number of users should be...
- Processing time or response time should be...
- TPS should be...
- Number of rows/records processed per minute should be...
- Bytes of data/second processed should be...
- Time to navigate form fields should be...

"What Do You Consider Acceptable Application Performance?"

(For example: 50 concurrent SQL*Forms 3.0 users, processing 1500 order forms a day from 9 am to 5 pm, with a 5 - 10 second response time for inserts and queries, and two SQL* ReportWriter users running large, query-intensive report activity on an intermittent basis.)

"How Much Oracle database and System Tuning Do You Expect to Perform?"

(For example: none, some initially, continuous with Oracle's help, extensive with Oracle training.)

Step 2. Define Your Customer's Application Environment

"Define Your Database."

- Total quantity of data will be...
- Database tables and their sizes will be...
- Frequently accessed/shared tables or data areas will be...

"Define Your Application."

If a full description of the application does not exist, describe the business needs driving application development:

- Co-existing non-Oracle applications include...
- The number of development and/or production users will be...
- Distinct Oracle applications (in-house, 3rd-party, Oracle) will include...
- Potential contention for data will be...
- Is this primarily a production or development environment, or a mixture?
- Percentage of work in each category will be...
- Describe batch processing during day shifts, during evening hours?
- Describe client-server activity on workstations?
 - Desktops?
 - Application?
 - Servers?
 - Over what protocol(s)?

Describe recovery...

- to disk or tape?
- Time limitations on recovery are...
- Will this system support training efforts?
- System should be available:
 - 5 days, 8 hours/day?
 - 5 days, 24 hours/day?
 - 7 days, 24 hours/day?
 - other?

"Define Your User Profile."

- Number of potential users will be...
- Average number of concurrent users will be...
- Will there be off-hour shifts?
- Categorize concurrent users into groups and describe activity:

| | | | |
|-----------------------------|-------|--------|-------|
| Interactive Production | heavy | medium | light |
| Batch Production | heavy | medium | light |
| Network (server) Production | heavy | medium | light |
| Interactive Development | heavy | medium | light |
| Network (server) Develop't | heavy | medium | light |
| Report Intensive | heavy | medium | light |
| Other? | | | |

"Define Your Transaction Profile."

Consider your business needs and describe a small set of average transactions (i.e. small, medium, large). Define:

- transaction complexity based on the SQL statements/application logic to be processed
- transaction units, for example, number of line items in an order
- transaction size, for example, number of rows read or written.

For example: "My average transaction consists of 4 selects, 3 updates, 2 inserts, 1 delete, 1 commit handling 3-8 line items processing 15-20 rows."

What response time do you expect for a small transaction?

- A medium transaction?
- Large?

"Define Your Reporting Activity."

Lengthy and/or complex reports can require significant CPU time and usually require a significant amount of memory and/or temporary disk space for sorts. Heavy reporting often interferes with the CPU, memory, and I/O use of other system activity, degrading performance for all users on the system.

What kind of reporting will be usual?

- Interactive ad-hoc queries?
- Reporting done in batch, day or night?

What will be the expected time interval reported on?

- Yearly?
- Quarterly?
- Monthly?
- Weekly?
- Daily?

How much data will be searched?
How much data will be returned?

“Define Your Batch Requirements.”

Because batch jobs do not interact with the user, they include virtually no wait time, for example, user think time. So, batch activity is typically transaction- or compute-intensive, placing a much heavier workload on the CPU than an interactive user. With “single-threaded” batch queues (1 batch job current at any time), heavy batch activity can consume up to 100% of one CPU. (On SMP machines, other CPUs will handle interactive activity.) With “multi-threaded” batch queues (>1 batch job simultaneously current), heavy batch activity can consume 100% of “c” CPUs where “c” is the maximum number of current batch jobs.

Will there be batch activity during the day?

Evening?

Both?

How often?

Batch activity will include...

Reporting?

Computational?

General Transactions?

Workloads will be:

Light?

Moderate?

Heavy?

Volume of data handled will be...

Describe batch transactions:

complexity

units

size

Have you considered single/multi-thread queues?

Step 3. Consider Other Application Issues:

Will there be any special processing at month-end? Quarter or year-end?

Will there be any regular conversion/loading of small or large tables?

Will there be any archiving/purging of tables? Small or large tables?

Will you perform mass deletes sequentially? Using indexes?

Will you regularly export/import to optimize database storage design?

Step 4. Consider Growth Plans

Will you add users (or clients) in the future?

How many?

Do you expect increases in transaction rates?

Will the database grow over time?

Table growth due to new data?

New index structures being added?

Historical data being stored?

Do you expect applications to support a growing number of new features/functionality?

Step 5. Determine which Oracle Products will be Used.

Consider multiple alternatives depending on your customer's preference and business needs:

SQL*Forms/SQL*Menu (for production and/or development)

Oracle*Card

Precompiled SQL

SQL*Plus

SQL*ReportWriter

CASE Products

SQL*Net

other?

Appendix B. Queuing Model Derivations

| | |
|-------|--|
| Z_i | is the CPU processing time for the i th transaction type (units are seconds). |
| t | is the CPU time to execute a particular transaction (units are seconds/transaction). |
| r | is the maximum possible arrival rate for a transaction using t seconds of CPU (units are transactions/second). |
| CPU | is the estimated CPU to support the i th transaction type (units are "CPU-units*seconds" per transaction.) |
| N | the number of users issuing the i th transaction (units are number of users). |
| n | the total number of transaction types considered. |
| K | is the SMP Factor that describes the physical number of "c" CPUs on the system (see SMP Factor K Table). |
| V_t | is total CPU requirements (units are "CPU-units"). |
| V | is the minimum uniprocessor CPU rating required to support the described workload (units are "CPU-units") on a system with "c" CPUs. |
| U | is the predicted system utilization based upon V_t . |

From M/M/c queuing theory (considering homogeneous transactions, in a single queue) where K is substituted for c here:

$$Z = \frac{1}{r(1-U)} \quad \text{since } r = \frac{1}{t} \quad \text{then } Z = \frac{t}{(1-U)}$$

$$\text{where: } U = \frac{V_t}{V * K} \quad \text{and } t = \frac{\text{CPU}}{V}$$

$$\text{so: } Z = \frac{\text{CPU}/V}{(1 - V_t/(V * K))} = \frac{\text{CPU}}{V(1 - V_t/(V * K))} = \frac{\text{CPU}}{(V - V_t/K)}$$

$$\text{so: } \frac{V - V_t/K}{Z} = \text{CPU}$$

for a single transaction "i":

$$V = \frac{\text{CPU}_i}{Z_i} + \frac{V_t}{K}$$

Considering multiple heterogeneous transactions (across multiple CPU's), we need to expand the single-transaction queuing formula to describe all the potentially concurrent transactions:

Since,

$$Z_1 = \frac{\text{CPU}_1}{V - V_t/K} \quad \text{or: } N_1 * Z_1 = \frac{N_1 * \text{CPU}_1}{V - V_t/K}$$

$$Z_2 = \frac{\text{CPU}_2}{V - V_t/K} \quad \text{or: } N_2 * Z_2 = \frac{N_2 * \text{CPU}_2}{V - V_t/K}$$

$$Z_n = \frac{\text{CPU}_n}{V - V_t/K} \quad \text{or: } N_n * Z_n = \frac{N_n * \text{CPU}_n}{V - V_t/K}$$

then,

$$N_1 * Z_1 + N_2 * Z_2 + \dots + N_n * Z_n = \frac{N_1 * \text{CPU}_1}{V - V_t/K} + \frac{N_2 * \text{CPU}_2}{V - V_t/K} + \dots + \frac{N_n * \text{CPU}_n}{V - V_t/K}$$

$$(N_1 Z_1 + N_2 Z_2 + \dots + N_n Z_n) = \frac{(N_1 \text{CPU}_1 + N_2 \text{CPU}_2 + \dots + N_n \text{CPU}_n)}{(V - V_t/K)}$$

$$(V - V_t/K) = \frac{(N_1 \text{CPU}_1 + N_2 \text{CPU}_2 + \dots + N_n \text{CPU}_n)}{(N_1 Z_1 + N_2 Z_2 + \dots + N_n Z_n)}$$

$$V = \frac{(N_1 \text{CPU}_1 + N_2 \text{CPU}_2 + \dots + N_n \text{CPU}_n)}{(N_1 Z_1 + N_2 Z_2 + \dots + N_n Z_n)} + \frac{V_t}{K}$$

therefore, for multiple transactions:

$$V = \frac{\text{SUM } i=1,n[N_i * \text{CPU}_i]}{\text{SUM } i=1,n[N_i * Z_i]} + \frac{V_t}{K}$$

RT_i is the total response time for the i th transaction type.

IO_i the number of physical I/Os done by the i th transaction (for sizing, estimate using the formula below).

IOT_i the I/O transfer time consumed by the i th transaction.

U_d an estimate of the average disk utilization based on the number of disks and the cache hit ratio.

CHR an estimate of the cache hit ratio (experiment with several values to examine the effect on the total response time).

To estimate the effect of the chosen system configuration on the actual total response time for a particular transaction, consider the impact of both CPU contention and disk contention on the transaction.

$$RT_i = Z_i + IOT_i$$

Use the steady-state formulas for the M/M/c queuing model for CPU contention and the M/M/1 queuing model for disk contention in series to account for both system resources.

$$RT_a = \frac{t_i}{(1 - V_t/(V_c * K))} + \frac{IO_i * (0.025 \text{ seconds per physical I/O})}{(1 - U_d)}$$

where $t_i = CPU_i/V_c$ and $IO_i = (CPU_i/0.009) * (1 - CHR)$

$$RT_a = \frac{CPU_i/V_c}{(1 - V_t/(V_c * K))} + \frac{(CPU_i/0.009) * (1 - CHR) * (0.025)}{(1 - U_d)}$$

Acceptable Response Time given by the customer can be described as:

$$RT = \frac{CPU_i/V}{(1 - V_t/(V * K))} + \frac{(CPU_i/0.009) * (1 - CHR) * (0.025)}{(1 - U_d)}$$

Because the values given for RT were used to determine V and the number of disks recommended (D) was based on $U_d = 25\%$, the ratio of the actual total response times versus the acceptable response times given by the customer is:

$$\frac{RT_a}{RT} = \dots$$

$$\frac{CPU_i/V_c}{(1 - V_t/(V_c * K))} + \frac{(CPU_i/0.009) * (1 - CHR) * (0.025)}{(1 - U_d)}$$

$$\frac{CPU_i/V}{(1 - V_t/(V * K))} + \frac{(CPU_i/0.009) * (1 - CHR) * (0.025)}{(1 - .25)}$$

which reduces to:

$$\frac{1}{V_c - (V_t/K)} + \frac{(0.025/0.009) * (1-CHR)}{(1 - U_d)}$$

$$\frac{1}{V - (V_t/K)} + \frac{(0.025/0.009) * (1-CHR)}{(1 - .25)}$$

$$= \frac{RT_a}{RT}$$

The ratio RT_a/RT described above is referred to in this document and the associated spreadsheets as X.

Appendix C. Uniprocessor CPU Ratings for VAX Models

| <u>VAX Model</u> | <u>VUP Rating</u> | <u>Number of CPUs</u> |
|----------------------|-------------------|-----------------------|
| <i>Older VAXes:</i> | | |
| 780 | 1 VUP | (non-SMP) |
| 8000 model 200 | 1.0 VUPs | (non-SMP) |
| 8000 model 250 | 1.2 VUPs | (non-SMP) |
| 8000 model 300 | 1.9 VUPs | (non-SMP) |
| 8000 model 350 | 2.3 VUPs | (non-SMP) |
| 8000 model 530 | 4.0 VUPs | (non-SMP) |
| 8000 model 550 | 6.0 VUPs | (non-SMP) |
| 8000 model 600 | 4.2 VUPs | (non-SMP) |
| 8000 model 650 | 6.0 VUPs | (non-SMP) |
| 8000 model 700 | ~6 VUPs | (non-SMP) |
| 8000 model 8x0 | 6.0 VUPs | (1 - 4 CPUs) |
| <i>Newer VAXes:</i> | | |
| 3100 model 30,40 | 2.7 VUPs | (non-SMP) |
| 3100 model 38,48 | 3.5 VUPs | (non-SMP) |
| 3100 model 76 | ~7.6 VUPs | (non-SMP) |
| 3100 model 80 | ~10 VUPs | (non-SMP) |
| 3100 model 90 | 24 VUPs | (non-SMP) |
| 3300 | 2.4 VUPs | (non-SMP) |
| 3400 | 2.4 VUPs | (non-SMP) |
| 3800 | 3.8 VUPs | (non-SMP) |
| 3900 | 3.8 VUPs | (non-SMP) |
| VAXft 3000 model 310 | 3.8 VUPs | (non-SMP) |
| 4000 VLC | 6 VUPs | (non-SMP) |
| 4000 model 60 | ~10 VUPs | (non-SMP) |
| 4000 model 90 | 24 VUPs | (non-SMP) |
| 4000 model 100 | 24 VUPs | (non-SMP) |
| 4000 model 200 | 5 VUPs | (non-SMP) |
| 4000 model 300 | 8 VUPs | (non-SMP) |
| 4000 model 400 | 16 VUPs | (non-SMP) |
| 4000 model 500 | 24 VUPs | (non-SMP) |
| 4000 model 600 | 32 VUPs | (non-SMP) |
| 6000 model 2x0 | 2.8 VUPs | (1 - 6 CPUs) |
| 6000 model 3x0 | 3.8 VUPs | (1 - 6 CPUs) |
| 6000 model 4x0 | 7.0 VUPs | (1 - 6 CPUs) |
| 6000 model 5x0 | 13.0 VUPs | (1 - 6 CPUs) |
| 6000 model 6x0 | 32.0 VUPs | (1 - 6 CPUs) |
| 7000 model 6x0 | ~36 VUPs | (1 - 6 CPUs) |
| 9000 model 110 | 40.0 VUPs | (1 CPU) |
| 9000 model 210 | 40.0 VUPs | (1 CPU) |
| 9000 model 3x0 | 40.0 VUPs | (1 - 4 CPUs) |
| 9000 model 4x0 | 40.0 VUPs | (1 - 4 CPUs) |
| 10000 model 6x0 | ~36 VUPs | (1 - 4 CPUs) |

Appendix D. Using the LOTUS 1–2–3 Spreadsheet

The sizing/capacity planning spreadsheet automates the calculations involved in the sizing and capacity planning methods presented in this guide.

Remember that the sizing estimates produced by these methods can be only as accurate as the data they are based on. Likewise, if the input to this spreadsheet is inaccurate, then the output of the spreadsheet will also be inaccurate. Therefore, output from this spreadsheet is not likely to be an absolutely accurate statement of the resources required to support a particular application, but rather a ballpark estimate of the required resources, roughly indicating the size and complexity of the hardware resources needed to meet your customer's performance expectations.

Step 1.

Copy the spreadsheet SIZVAX2.WK1 to a temporary spreadsheet, TEMP.WK1, so you do not accidentally modify or lose the original.

Step 2.

Review Part I of this guide, and complete the customer survey in Appendix A.

Step 3.

Use the information gathered from the survey questions to define general transaction types in the customer's application. Define various user classes based on the transactions that users employ.

For example, a moderate SQL*Forms transaction might consist of:

- making several complex queries through a moderately complex form/views
- updating or inputting multiple column values for multiple rows through a simple form
- querying those rows out of the table(s) for verification using the initial, moderately complex form.

Define your user classes based on which transactions types a user employs most often.

Step 4.

When sizing:

When you have defined several transaction types, find a transaction type on the spreadsheet that corresponds to each of your transaction types. Place each user in the category which best represents his/her most commonly executed transaction.

For example, a data entry operator may employ several different transaction type including:

- Two simple, light-weight forms
- Three moderately heavy forms
- One very heavy form

If the user uses the moderately heavy forms more than 60 - 80% of the time, classify the user as a "Moderately Heavy Forms User" in column C, **Number of Users**.

There are hidden columns F and G, **CPU per Transaction** and **Memory/User** respectively, on the spreadsheet which contain typical "CPU used per transaction" and "memory per user of that transaction" values for each generic transaction type on the spreadsheet, expressed as "CPU-units *.seconds" and "MB/User". These values are "weights" collected empirically from various tests and installed customer sites to provide default values for various generic transaction types. (These are the values from the "Sizing Tables".)

However, these weights are not absolutely accurate values. They merely represent typical transactions which would fall into the generic transaction categories on the spreadsheet. If you are not certain which category applies to your particular transaction type, compare values from a more complex transaction type and from a less complex transaction type, and extrapolate to arrive at values which represent your transaction.

Then, in any row, replace the hidden columns F and G values in the spreadsheet with your new "CPU per Transaction" and "memory per user" values. The spreadsheet will use the new values in all the dependent calculations. **DO NOT SAVE THE SPREADSHEET WITH THE ALTERED F and G COLUMN VALUE(S).**

Hidden column H, **LIO per transaction**, contains a simple formula to estimate the number of Logical I/O's per transaction, based upon the CPU value in column F and the CPU/LIO correlation factor. The result of this formula is used to calculate the IOT value in column K.

When planning capacity:

Since the application already exists, **simply replace the values in the hidden columns F, G, and H with the CPU, memory, and LIO values (respectively) that you measured and collected on your test system.**

Note: Be certain to normalize the CPU values by multiplying the CPU time used by the uniprocessor CPU rating of your test CPU system. Do not use the total CPU rating of the test system if it has multiple processors (CPUs).

When sizing or planning capacity:

At this point, each expected user should be accounted for on the spreadsheet in the second column C, "Number of Users", but in only one of the transaction types. Do not associate any user with more than one type of transaction. This would falsely increase the Estimated Memory Requirements value at the end of the spreadsheet.

Step 5.

For each transaction type, estimate how often a typical user will employ the transaction (transaction/second/user). Higher frequency will result in higher CPU requirements; lower frequencies will result in lower CPU requirements. Be conservative but do not under-estimate the speed at which end-users will attempt to process data. Place these values into the third column E, **Transaction Frequency**.

Step 6.

Now define a measure for your customer's performance expectations. This measure of performance expectation is the total response time that your customer finds acceptable: *the total amount of time needed to process all the transaction components minus "think time" and typing time.*

Enter an acceptable total response time (really the maximum allowable total response time) in the fourth column I, **Acceptable Response Time**, for each transaction type on the spreadsheet. You should already have gathered values for the Acceptable Response Times column when you interviewed the customer.

Note: Examples of acceptable total response times may already have been included with the spreadsheet. Do not depend upon these values as accurate. They are intended only as examples and will affect the output.

Note: Your Total Acceptable Response Time must be greater than the value for the estimated I/O Transfer Time (IOT) in column K to the right. **If any of the Response Time values are less than the respective IOT value calculated by the spreadsheet, the spreadsheet "V" output will be meaningless.** Since the IOT value is highly dependent upon the number of disks and the average disk utilization, you may want to return to this step once you have selected a number of disk drives in the next step.

Do not enter values into the IOT column. IOT values are calculated by the spreadsheet based on the LIO values in the hidden column H, the value entered for the expected cache hit ratio, and the estimated disk utilization. (**Adjust the number of disks [ie. to lower the disk utilization] or the cache hit ratio accordingly to reduce the calculated IOT values.**)

Step 7.

You should now have entered values for the first three columns: Number of Users, Transaction Frequency, and Acceptable Total Response Time. The spreadsheet will have automatically calculated values for the total Memory Requirements and the total CPU Requirements columns.

The minimum memory required to support only the users (not including the SGA) will be displayed in the first line of the lower section of the spreadsheet. The estimated CPU requirement is not displayed on the spreadsheet but is used in a queuing model to determine a minimum recommended uniprocessor CPU rating which includes consideration for the performance criteria (i.e., the acceptable total response

time values). This value is displayed further down in the spreadsheet and depends on the number of CPUs chosen for the system.

Step 8.

Testing has demonstrated a loose correlation between the CPU a transaction uses and the number of logical I/Os that same transaction performs. This correlation is discussed in Step 3 of this document. (The number of logical I/Os represents the number of times that the Oracle Server needed to access a disk. The number of physical I/O's is the actual number of times that the Oracle Server did access a disk, that is, the request could not be satisfied by reading data out of the SGA. The cache hit ratio is the percentage of logical I/Os that did not result in physical I/O.)

Now choose a value for the expected **cache hit ratio** (CHR). The spreadsheet uses this value with the LIO values from the hidden column H above to estimate a total Physical I/O rate (PIO) for all the users of this application. The PIO rate will be used to recommend a number of disk drives. Requiring the CHR to be input forces the customer to consider the effort spent tuning and tailoring this environment.

Enter several different values for the cache hit ratio and observe their affect on the total physical I/Os the system disks will need to support. Observe also the number of disks recommended to support the physical I/O load. This recommendation is based upon the value you enter for the "**optimal I/O rating**" for your type of disks. The default, accurate for most Digital disk drives, is 40 I/Os per second.

Higher I/O rates will eventually cause disk request queues to build up, increasing I/O wait time, increasing the total time to process each I/O request, and subsequently the total response time.

Step 9.

Choose a number of disks for the system. The spreadsheet uses the I/O throughput capacity which you entered above (default of 40 I/Os per second) to estimate the **average utilization** of each disk drive. This value is presented to demonstrate the worst-case impact of under-sizing the number of disks on the system. If the estimated disk utilization is greater than 100%, add more disks to the configuration.

Note: Under-estimating the number of disks necessary for optimal performance is a very common cause of performance problems. Any increases in I/O processing times will translate directly into increases in user response time.

The average disk utilization is used in the simple M/M/1 steady state formula in column K to determine IOT values.

Step 10.

Next, choose the number of CPUs on the planned system. You can optionally enter an SMP scalability factor for this type of computer platform. There is a default value of 75% for the SMP scalability field. Entering either of these value will cause the spreadsheet to recalculate the minimum recommended uniprocessor CPU rating value in the next row. Try several different numbers of CPU's (from 1 to 6) and observe how the minimum suggested uniprocessor CPU rating alters based on the number of CPUs. Then decide on an appropriate number.

If the minimum recommended uniprocessor CPU rating becomes less than 0 or inordinately large (several 100), one or more of the IOT values is larger than its respective Acceptable Response Time value. Either adjust the Acceptable Response Time values accordingly; or increase the value entered for the cache hit ratio; or increase the number of disk drives to lower the disk utilization. All will decrease the calculated values for the IOTs.

Note: SMP systems (multiple CPUs in one computer) typically benefit environments which require batch activity during typical operating hours (i.e. 9:00am to 5:00pm).

Step 11.

Now that your minimum uniprocessor CPU rating is determined, choose an actual uniprocessor CPU rating for a particular system. (Refer to Appendix C.)

The spreadsheet will then estimate what percent of the total CPU available (100% includes all CPU's on the system) will be utilized by the workload according to the users/transactions described in the spreadsheet.

Note: If the estimated utilization value displayed is greater than 100%, choose another system with a larger uniprocessor CPU rating, or, if you chose an SMP system, increase the number of CPUs selected. **CPU Utilization above 100% will render the spreadsheet output meaningless.**

Step 12.

The impact of the system's estimated CPU utilization is displayed near the bottom of the spreadsheet. This factor represents an estimate of how actual worst-case CPU processing times will compare to the acceptable CPU processing times calculated from the Acceptable Response Time column.

Negative values displayed in this field are meaningless and indicate only that the estimated CPU utilization is still greater than 100%. To correct this, choose a larger CPU or more CPUs and decrease estimated CPU utilization below 100%.

Step 13.

The impact of the system's estimated disk utilization is displayed near the bottom of the spreadsheet. This factor represents an estimate of how actual worst-case disk processing times will compare to non-loaded disk transfer times.

Negative values displayed in this field are meaningless and indicate only that the estimated disk utilization is greater than 100%. To correct this, choose more disks or increase the cache hit ratio estimate you entered earlier.

Step 14.

On the bottom row of the spreadsheet there is a factor, X, which represents the combined impact that the estimated CPU and disk utilization could have on the worst-case total response times for end-users. This value represents the "average" impact, under peak load, across all transactions. Values in this field greater than 1.00 do not necessarily indicate a problem, but do indicate that under *peak* end-user workload (as described in the table at the top of the spreadsheet), the estimated worst-case total response times may be greater by this factor than the customer's acceptable response time requirements (as input at the top of the spreadsheet). For the impact on specific transactions, see the rightmost column (T) in the top section of the spreadsheet. Values less than 1.00 also do not indicate a problem, but may suggest the chosen configuration is oversized.

For example, if the ratio is 1.40, then the worst-case total response time may be up to 1.40 times longer than the Acceptable Total Response Time values. Try to provide a configuration that results in a value as close to 1.00 as possible.

| Under Load: | Number of users: | Transaction Frequency: 'R' txn/hr/user | Acceptable Response Time: (must be > than) | | CPU SUM | MEM SUM | LIO/sec SUM | Response Time Peak seconds |
|--------------------------|------------------|--|--|-------|------------|---------|-------------|----------------------------|
| | | | 'RT' seconds | 'IOT' | | | | |
| ===== | | | | | | | | |
| SQL*FORMS 3.0 (runtime) | | | | | | | | |
| Heavy, complex form | 0 | 0.00 | 80.00 | 21.98 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately heavy form | 0 | 0.00 | 40.00 | 11.57 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderate form | 0 | 0.00 | 20.00 | 7.33 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately light form | 0 | 0.00 | 10.00 | 3.47 | 0.00 | 0.00 | 0.00 | 0.00 |
| Light form | 45 | 92.00 | 7.00 | 2.70 | 40.25 | 123.75 | 894.44 | 3.11 |
| Total no. forms users | 45 | | | | 87.50 | 232.50 | 1944.44 | |
| | | | | | SPECfp92's | MB's | LIO/sec | |
| ----- | | | | | | | | |
| SQL*REPORTWRITER | | | | | | | | |
| Heavy, complex report | 0 | 0.00 | 120.00 | 88.67 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately heavy report | 0 | 0.00 | 80.00 | 52.05 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderate report | 0 | 0.00 | 40.00 | 28.92 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately light report | 0 | 0.00 | 25.00 | 15.42 | 0.00 | 0.00 | 0.00 | 0.00 |
| Light report | 0 | 0.00 | 12.00 | 7.33 | 0.00 | 0.00 | 0.00 | 0.00 |
| Total no. report users | 0 | | | | 0.00 | 0.00 | 0.00 | |
| | | | | | SPECfp92's | MB's | LIO/sec | |
| ----- | | | | | | | | |
| SQL*PLUS | | | | | | | | |
| Heavy, complex trans | 0 | 0.00 | 110.00 | 79.04 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately heavy trans | 0 | 0.00 | 50.00 | 34.70 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderate trans | 0 | 0.00 | 25.00 | 16.58 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately light trans | 0 | 0.00 | 15.00 | 9.64 | 0.00 | 0.00 | 0.00 | 0.00 |
| Light trans | 0 | 0.00 | 5.00 | 2.70 | 0.00 | 0.00 | 0.00 | 0.00 |
| Join on EMP and DEPT | 0 | 0.00 | 0.50 | 0.15 | 0.00 | 0.00 | 0.00 | 0.00 |
| Total no. SQL*PLUS users | 0 | | | | 0.00 | 0.00 | 0.00 | |
| | | | | | SPECfp92's | MB's | LIO/sec | |
| ----- | | | | | | | | |
| PRO*C | | | | | | | | |
| Heavy, complex trans | 0 | 0.00 | 2.25 | 1.54 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately heavy trans | 0 | 0.00 | 2.00 | 1.47 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderate trans | 0 | 0.00 | 1.00 | 0.77 | 0.00 | 0.00 | 0.00 | 0.00 |
| Moderately light trans | 0 | 0.00 | 0.25 | 0.10 | 0.00 | 0.00 | 0.00 | 0.00 |
| Light trans | 0 | 0.00 | 0.25 | 0.08 | 0.00 | 0.00 | 0.00 | 0.00 |
| Extra light (TPC-B) | 0 | 0.00 | 0.25 | 0.07 | 0.00 | 0.00 | 0.00 | 0.00 |
| Total no. PRO*C users | 0 | | | | 0.00 | 0.00 | 0.00 | |
| | | | | | SPECfp92's | MB's | LIO/sec | |
| ----- | | | | | | | | |

```

Estimated minimum memory required          123.75 MB's
  (for USERS only, no SGA)

  ENTER the expected cache hit ratio:      —> :  0.90
                                           ----

Physical I/O rate speculation:             75    to    112 IO/sec

  ENTER the optimal I/O rating  (IO's per second) —> :  40
                                           ----
  for these disks:

Recommended number of disk drives:         7    to    11 disks
  (based upon PERFORMANCE considerations only; ~25% thrupt utilization)

  ENTER the expected number of disks:      —> :  10
                                           ----

Estimated disk I/O utilization:            19    to    28 %
  (assuming well balanced I/O rate)

  ENTER the number of CPU's                —> :  1
  (ie. 1 = uniprocessor)                  ----

  ENTER the SMP scalability factor for this system —> :  0.75
  (between 0.00 - 1.00; ie. .75 = 75%)    ----

Minimum recommended UNIProcessor SPECfp92's rating: 48 SPECfp92's
  for a system with          1 CPU's

  ENTER the UNIProcessor SPECfp92's       —> :  126
  rating for a selected system            ----

Estimated CPU utilization for a           1 CPU   :  32 %
Alpha AXP OpenVMS system having a
UNIProcessor rating of 126 SPECfp92's

On average, estimated CPU processing times under PEAK load may be 0.09 times
the allowable CPU processing times for the response times given

On average, estimated I/O processing times under PEAK load may be 1.39 times
the non-loaded disk transfer times

On average, estimated RESPONSE times under PEAK CPU and DISK load may be greater
than the Acceptable Response Times given by a factor of 0.45
(NOTE: SEE COLUMN 'T' for estimated response times under PEAK load.)

```

ORACLE®

**Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
415.506.7000
800.633.0676 ext.13
Fax 415.506.7200**

**International inquiries:
44.932.872.020
Telex 851.927444 (ORACLE G)
Fax 44.932.874.625**

**Copyright © Oracle Corporation 1994
All Rights Reserved
Printed in the U.S.A.**

Part A19176