NOTICE

COMPAQ TECHNOTE

CONFIGURING COMPAQ RAID TECHNOLOGY FOR DATABASE SERVERS

First Edition (July, 1994)
Part Number 184206-001

CONTENTS

ARRAY CONFIGURATION GUIDELINES (continued)

Appendix A
MULTIVENDOR INTEGRATION FROM COMPAQ

Chapter 1
INTRODUCTION

This document provides general guidelines when determining the best configuration of the Compaq SMART SCSI Array Controller for your database server needs. Although this document cannot predict absolute performance, it provides the necessary information to make informed configuration choices given your performance, fault-tolerance, and cost constraints.

This document presumes that the reader has a general familiarity with:

- The fault tolerance model called RAID (Redundant Arrays of Inexpensive Disks).
- Compaq drive subsystem terminology.
- The use of Compaq System Configuration utilities.

DRIVE SUBSYSTEM PERFORMANCE

Drive subsystem performance is one of the most important aspects of tuning a database server for optimal performance. Compaq has designed drive controllers that utilize RAID technology to provide businesses with the best balance of performance, reliability, and availability. RAID technology provides a system with more consistent performance, higher levels of fault-tolerance, and easier fault-recovery than traditional non-RAID systems.

Although many artificial benchmarks show non-RAID systems outperforming RAID systems, Compaq has not seen a case of this being true in a production environment. The mechanical nature of the disk drives is the basic limiting factor for I/O performance. Therefore, efforts to configure and tune a system should focus on getting the most out of each drive and having an appropriate number of drives on a system. When properly configured, the I/O subsystem should not be the limiting factor in overall system performance.

REFERENCE MATERIAL

Refer to the following Compaq publications for information on the SMART Controller and drive array diagnostics:

- *Compaq SMART SCSI Array Controller User Guide* (part number 142136-003)
- *Drive Array Advanced Diagnostics User Guide* (part number 137331-001)
- *Drive Array Advanced Diagnostics Addendum to User Guide* (part number 137563-001)

Chapter 2
DATABASE I/O OVERVIEW

You need a basic understanding of database access patterns and methods to configure the drive subsystem of a database server correctly. This topic provides a high-level overview of the database process and typical patterns of common application types.

DATABASE I/O SCHEMES

Database systems store data in a logical collection of disk files or disk partitions. The physically atomic unit of a database file is called a *database page*. The typical size of a database page is 2 Kbytes. Most databases do not allow the user to change this size. Oracle does, however, allow the administrator to define the page size at database creation time. The default page size of Oracle for NetWare is 4 Kbytes and the default page size of Oracle for UNIX, Microsoft NT, and OS/2 is 2 Kbytes. Changing the page size from the default has rarely demonstrated any measurable performance gains.

Because the database page handles the physical reads and writes of the data, disk requests are usually 2 Kbytes. There are some exceptions to this on large table scans. If the database system detects that a long scan will be done it often reads in larger blocks. However, this feature does not usually change the tuning process.

The process of executing I/O routines is similar across most database systems. Database process(es) or thread(s) for the application perform reads of database pages. Many read operations can usually be conducted in parallel and independent of each other. Writes of data pages, on the other hand, are not initiated by a user application. When an application updates or inserts new data, the change is put into the database cache by the application's database process. Some time later an independent database function cleans the dirty data pages from memory. These writes usually occur as a large batch of asynchronous writes.

The initiation and frequency of this process is beyond the scope of this paper. Write operations are not usually a steady stream of single-page writes but are periodic, heavy bursts. In contrast to the data pages, writes of transaction log pages is a very steady stream of single-page writes. A background database process controls these. They occur during update and insert operations.

APPLICATION TYPES

You can determine the fundamental I/O profile used to access the disk by examining different database applications and database functions. This determines whether a particular dataset is accessed sequentially or randomly. Key high-level application types include transaction processing, decision support systems, and batch processing. The related topics listed below examine the various access patterns on the database files, including data and index files, transaction logs, and import/export files.

On-Line Transaction Processing

On-Line Transaction Processing (OLTP) is usually characterized as many users acting on a small subset of data throughout the database. The I/O profile resulting from this load is very heavy random reads and writes across the data and index files. The transaction logs, however, are hit with a heavy stream of sequential write operations of 2 Kbytes or less.

Decision Support System

Decision Support System (DSS) is characterized by multiple users executing complicated joins and aggregations on large sets of data. Even though many of the operations could use some sequential processing, contention with other users and join and indexing operation result in a fairly random access pattern to the data and index files. Usually, if the database is dedicated to DSS no updates are performed to the database during heavy query load. In this case, no I/O occurs on the transaction logs. If on-line updates are applied to the database, log file response time can still be important to the overall performance of the system.

Batch Processing

Batch processing is the most likely application to produce significant sequential I/O on the data files. These activities often occur after hours and usually in isolation of other activities. Batch processing involves database dumps, database loads, detail report processing, and index creation.

Chapter 3
ARRAY CONFIGURATION GUIDELINES

The key issues when configuring a drive array system are how it is configured into logical volumes and how many physical disk drives are in the volumes. This chapter discusses these issues.

VOLUME CONFIGURATION

The most difficult decision to make when configuring a system is how to partition an array of physical drives into logical volumes. This can be a very easy step in the configuration process, once the methodology is understood. The key is to let an array do what it is intended to do, that is, distribute I/O requests among all available drives.

The basic principles involved are as follows:

- Group all randomly accessed data on a single volume.
- Isolate purely sequential I/O to avoid excessive drive head movement.

Group Randomly Accessed Data

Most data access on database servers is random. When your goal is to optimize these application classes, group the database data and index files on a single-array volume for each drive array controller. For example, if an OLTP database contains a number of large tables and their associated indexes totaling 5 Gigabytes, put these on a single volume of six 1-Gigabytes drives. This ensures that all drives have approximately equal I/O load during normal operation of the system. The drive controller performs this load balancing by performing a low-level striping of files across all disks in the volume. There is no additional effort required by the System Administrator or the Database Administrator (DBA).

In contrast, techniques that have been carried over from traditional, non-array systems include the separation of data and index files to separate physical drives and separation of large data tables to their own dedicated drives. This allows simultaneous access to multiple drives. The problem with these techniques is that it puts the burden of load balancing upon the System Administrator.

By carefully studying the applications accessing the database, the data can be distributed so each drive receives an approximately equal number of disk requests. Unfortunately, two assumptions are made that are difficult to achieve. These are that the System Administrator and DBA can determine the actual I/O profile of an application and, that once determined, the I/O profile remains constant throughout a day's processing and over the life of the application. For example, an application often uses different tables at different times of the day. Table A might have heavy access in the morning as a business gets started for the day. As the day wears on, most access might move over to Table B. If these tables are on different physical disks, the application would overload the disk with Table A in the morning and overload the disk with Table B later. However, an array controller places these two tables on the same logical volume spread across multiple drives allowing all drives to share equally in the I/O load throughout the day.

Another example to further demonstrate this concept is data growth, again using Table A and Table B. When the database is initially deployed Table A and B are approximately the same size and receive the same number of I/O requests. At that time, it makes sense for each table to be on its own drive. Over time, Table B grows by 10 percent per month in both size and I/O requests. After just 9 months Table B has doubled in access frequency and is now a bottleneck to system performance. If both these tables are put on a single array volume, this table growth is shared among all drives on the volume, thus avoiding the disk bottleneck.

The case covered thus far applies when all drives are attached to a single controller. When the amount of data and indexes exceeds the capacity of a single controller you will have multiple volumes in the system. At this point you have three options for distributing the data across the controllers.

- Use an operating system facility to stripe the volume across controllers.

- Use the database facilities to stripe the data across the controllers.
- Segment different data to each controller.

There is no *best solution* to this situation. It is a matter of preference to the Administrator. The degree of each of these can vary depending upon operating system and database software you choose. The following table lists pros and cons of each option.

Table 3-1
Data Distribution Options

Option	Pros	Cons
Use an operating system facility to stripe the volume across controllers.	Ease of space management as database grows Ease of backup with everything on one file system	More difficult volume management Very long tape recovery times of volume Performance penalty associated with file system striping
Use the database facilities to stripe the data across the controllers.	Little to no negative performance impact Ease of space management as database grows	Most complicated database configuration process More complicated backup/recovery process
Segment different data to each controller.	Best performance when balanced properly Fastest, easiest tape recovery of volume Fastest tape backup, allows multiple tape usage	Requirement of administrator to understand access patterns

Isolate Sequentially Accessed Data

Even though most data in a database server is read and/or written randomly, there are several cases of sequential access. If the drive head can stay on the same physical disk track during a long sequential read or write process, performance can be dramatically better than if the head must continually seek between tracks. This principle is the same for traditional, non-array controllers and array controllers.

The largest, single time component in reading or writing a block of data on a disk is the time required to move the drive head between tracks. The best drives on the market today range from 8 to 10 milliseconds (ms) average seek time. Many drives you might be using could have average seek times 16 or 19 ms.

The remaining sequence activities in an I/O routine are as follows:

1. The database makes a request from the operating system.
2. The operating system requests the data from the controller.
3. The controller processes the request and makes the request from the drive.
4. The block rotates under the drive head.
5. The data is transferred to the controller.
6. The controller transfers the data back to the operating system and then to the application.

All of this takes 10 to 15 ms, depending upon the particular controller, operating system, and database. Minimizing seek time, when possible, has a significant benefit.

What does this mean to your application and how do you take advantage of it? There are three basic database operations that are characteristically sequential: transaction logging, bulk loading and unloading of the database, and batch reporting or updating requiring table scans. In many cases involving sequential I/O, the operations associated with it are not the primary function of the system but might be the most important aspect for your tuning due to nightly batch windows or backup time constraints. If the typical random access to the data is relatively light or done by few users, you might want to give priority to tuning sequential operations over random I/O distribution.

The transaction log is the most obvious sequential I/O routine. In any update environment, the transaction log is sequentially written from beginning to end. Isolate the log file from any other system I/O activities during heavy update activity. However, a typical log file's space requirement might not need the space of an entire physical disk. You can usually share a single large disk between the log file and other files on the system which are not active during heavy update activities. Examples of items which coexist well with the log file include the operating system, database executables, database dump files, and so on.

Bulk loads and batch updates are often associated with decision support systems (DSS). The typical scenario involves extracting large volumes of data from production transaction processing systems on a pre-defined schedule, such as daily or weekly. The data is then moved to the DSS server and loaded into a database. Performance of this load operation can be very critical due to nightly time constraints.

For example, presume you have a decision support system which is refreshed nightly. The following table gives application characteristics and an optimal configuration for loading the database.

Table 3-2
Database Loading Characteristics

Data File	Space Requirement	Disk Volume	Drive Count	Drive Size
Database Log and Operating System	500 MB	Volume 1	1	1 GB
Database Data File	2.5 GB	Volume 2	3	1 GB
Database Index File	1 GB	Volume 3	2	1 GB
Extract File	1.5 GB	Volume 4	1	2GB

The final general application classes which rely on sequential I/O performance include batch reporting and database dumps. These are fairly simple operations to optimize and there are only a couple of concepts to keep in mind. The profile of reporting and unloading is sequential table scans with some resulting output being written to a non-database file. In the case of a detailed analysis report, a report file of several megabytes may be spooled to a large disk file. The database dump sequentially reads tables and then writes the results to either a tape device or disk file. When the output is written to a disk file on the server, the target disk should be a different physical volume from the source of the data. If this is the only activity at the time, the volume used for the log file is a good target. If this is not the only activity, add another volume to the system of sufficient size to accommodate the output file.

Optimizing this process is very different from the typical system. In contrast to the random nature of most multiuser systems, a batch load or update can have up to 4 concurrent sequential operations. These operations include reading the input file, writing the database data file, writing the database index file, and writing the transaction log. To optimize the load time of the data, isolate each of these file types to their own logical volumes. In contrast, pure optimization for daily multi-user decision support would put ALL of the above files on a single, large volume to allow concurrent access of all drives on the server by all read operations. The performance impact on daily access of optimization for load times should be less than 20 percent. However, the gain in load time can be over 200 percent which is often well worth the small, random-access penalty.

FAULT-TOLERANCE EVALUATION

There are several fault-tolerance options to a System Administrator depending upon the integrity requirements, the availability requirements, and the recovery issues of the application being deployed. The Compaq SMART SCSI Array Controller RAID levels relevant to this discussion are RAID 0, RAID 1, and RAID 5. RAID 0 refers to data striping multiple disks into a single logical volume and has no fault tolerance. RAID 1 and RAID 5 are both fault-tolerant, but require a different amount of drives to accomplish the fault tolerance. RAID 1 or drive mirroring uses 50 percent of the drives in a volume for the fault tolerance. RAID 5 requires only a single drive worth of the volume's total capacity for the fault tolerance. For example, if a RAID 5 volume is configured with six 1-Gigabyte drives, the usable capacity of the volume is 5 Gigabytes. With RAID 1 this same 5 Gigabytes of usable space requires ten 1-Gigabyte drives.

The choice of RAID is dependent on the individual situation. No one alternative is best for every application. Some applications might need complete redundancy and the System Administrator is willing to pay for it. Other System Administrators might be willing to risk some down time for a more cost-effective implementation. In addition, I/O loads generated by a particular application might fit very well in the performance constraints of RAID 5 while other applications suffer serious performance penalties with RAID 5.

The related topics explain some of the trade-offs associated with each alternative and possible performance implications. However, you must evaluate each alternative's applicability to your environment.

Data Protection Levels

Data protection refers to the degree the database integrity is protected from a physical disk failure. The chosen scheme is frequently determined by the amount of down time or data loss your organization can afford. As the degree of protection increases, so does the cost of the system. However, the cost of a few thousand dollars of redundant disks might be much less than the cost of a system being down.

The most secure option is to configure the entire drive subsystem to run in a fault-tolerant mode. In this case, no single disk failure stops the system or risks any loss of data. If the drive subsystem supports hot-swap drives, such as the Compaq SMART SCSI Array Controller with a Compaq ProLiant Server, the system need not be brought down for repairs. If the system does not support this feature, you must schedule down time. Most business critical applications, particularly OLTP systems, use this level of fault-tolerance.

Another option uses fault tolerance on some volumes, but leaves others unprotected from a single disk failure. This involves putting all database transaction logs, critical database system tables, the operating system, and boot partition on a volume with some form of fault tolerance. The remainder of the system is configured with volumes using no fault tolerance. These volumes contain general data tables, index structures, and other non-critical database objects such as temporary space.

If a protected drive fails, the system continues to run in the same manner as describe in the previous configuration. If an unprotected drive fails, the system eventually fails and the current instance of the database is lost. At this point the system is brought down and repaired, the database is restored from a previous backup, and the database restored to the instant of failure by rolling forward committed transactions from the protected transaction logs.

Protecting the operating system and system tables of the database speeds up the recovery process. If you lose the operating system or system catalogs, you can still recover the database. However, the process is more complicated and takes much longer.

A third configuration option is to run with the disk subsystem completely unprotected. In this situation, any disk failure usually results in the requirement to restore the full database from a previous backup or reload the database if loaded from an ASCII dump of another system. All updates performed since the backup or original load are lost and are *not* recoverable. This might be acceptable for many types of operations such as static weekly loads of a decision support system.

RAID Level Considerations

To protect a database from a drive failure, you can configure the Compaq SMART Controller volume with either RAID 1 or RAID 5. Compaq array controllers allow you to mix RAID levels on a single controller when system requirements need varying level of performance and protection.

No single configuration fits all application scenarios. This discussion gives a System Administrator or Database Administrator the information needed to make an educated decision about RAID configurations, depending on an application's specific characteristics.

COST/PERFORMANCE TRADE-OFFS

In general, RAID 5 provides a more cost-effective solution compared to RAID 1, particularly when I/O performance requirements are well below the performance capabilities of the configured drives. The cost advantage becomes more pronounced as the capacity requirements increase. For example, in a configuration needing only 1 Gigabyte, a RAID 5 solution could be three 550-Megabyte drives and the RAID 1 solution would be two 1-Gigabyte drives. There is very little difference between these two configurations' prices. However, if you are looking at a 14-Gigabyte system, you could configure RAID 5 with eight 2-Gigabyte drives as opposed to fourteen 2-Gigabyte drives for the RAID 1 configuration.

Even though RAID 5 often has a cost advantage, RAID 1 has its own advantages for price/performance and increased fault-tolerance. If the application(s) using a database server have high I/O requirements, RAID 1 provides significantly better performance at the same user capacity due to the increased number of drives. Increasing the performance of the RAID 5 volume requires increasing the number of drives on the volume and the capacity over what the application needs. RAID 1 and RAID 5 can have the same price/performance costs on high I/O systems.

Another advantage of RAID 1 is its ability to tolerate more than a single drive failure. If drives fail, the server continue running as long as the multiple failed drives are not the pair mirroring each other. For example, in a RAID 1 volume consisting of 4 drives mirrored to 4 drives, the first drive of the first set of drives fails. Before that drive is replaced, any other drive (except for the first drive in the second set) could also fail and the system would continue to operate. In contrast, if a single drive fails on a RAID 5 volume and then a second drive fails before the first drive is replaced and recovered, the data is lost.

FAILED STATE CONSIDERATIONS

The only configurations where this is relevant is RAID 1 and RAID 5. The performance degradation for RAID 1 should be negligible because the only difference is that one less drive is available for read operations. However, RAID 5 can incur a more significant penalty. The reason for the degradation is that for every read or write request to the failed drive, an I/O operation against all other drives in the volume is required. In most database systems the sustainable load drops until you replace the drive and complete the rebuild process.

RECOVERY TIME

You need approximately 20 to 30 minutes per Gigabyte to recover from a drive failure if the system is idle. If the system is in use during the drive rebuild, recovery time can be dependent on the level of activity. Most systems should recover in nearly the same time with moderate activity as with no load, particularly RAID 1. RAID 5 is much more sensitive to system load during the recovery period due to the considerably heavier I/O requirements of the failed system. Consider this recovery time and failed-state performance when deciding between RAID 1 and RAID 5.

PHYSICAL DISK REQUIREMENTS

Once you understand the target environment, the volume configuration guidelines, and the fault-tolerance requirements, the next step is determining how many disks to distribute the data across and which type of disks are required. The goal is to use the drives with the lowest cost per megabyte (usually the largest drives available) while having enough drives to satisfy the physical I/O rate required by the application.

The number and type of drives required by a disk subsystem volume depends upon the throughput requirements of the environment and the total amount of data on the server. Throughput requirements are composed of two characteristics, the degree of concurrency among user requests and the request rate from the application. Concurrency and request rate are distinct, but related concepts. Concurrency refers to the number of physical requests that the drive controller(s) are executing. The degree of concurrency is primarily influenced by the number of users on the system and the frequency of requests from those users.

The request rate is the absolute number of reads and writes per unit of time, usually expressed as I/Os per second. Usually, the higher the concurrency level, the higher the request rate. Yet, fairly high request rates can be generated by a single batch process with no concurrent requests. Of course, the number of physical disk requests that occur depends upon the size of the database relative to the amount of disk cache and the resulting cache hit rate. The worst-case scenario would result in as many concurrent disk requests as simultaneous user requests.

The reason for understanding the application I/O request rate is that multiple I/O requests can typically be processed in parallel if the data is distributed across multiple disk drives. Therefore, even though large drives are usually "faster" than smaller drives, this performance difference cannot offset the advantage of two drives responding to multiple requests in parallel. The target number of drives in a server should be approximately equal to the sustained or average number of simultaneous I/O requests, with three drives being a typical minimum for the data and index files.

Evaluation Process

The following evaluation process determines the number of drives required by a volume containing data accessed randomly by multiple users. Repeat this process for each volume to be used on a database server. In the case of high transaction rate systems, this process might yield too many drives to fit on a single controller. At this point the drives can be split into multiple, equal volumes across controllers. For systems or volumes using purely sequential I/O, or having no concurrent disk requests with only one or two users on the system at a time, strict calculations for the number of drives is not required. For this case, use the fastest drives available and as many as required to meet the space requirements of the system.

The procedure to determine the optimal number of drives is summarized here followed by a detailed explanation and several case studies:

1. Determine typical I/O request rate by analyzing transaction I/O requirements and typical transaction rates of the target application(s).
2. Determine the minimum number of drives required to fall within or below the 35 to 45 I/Os per second rates of typical disk drives.
3. Add at least 20 percent space for growth and administration to the database size.
4. Divide drive requirements based on RAID usage into total space requirements to determine which drive size to use.

The first step in calculating the number and size of drives to use is determining the request rate of the application. Without this information you cannot make an accurate estimation of the number and type of drives to use.

The following equations provide rough estimates about the number of I/O requests generated by an application. Due to optimization routines in the controller to reduce the total I/O requests, these equations do not yield exact loads. They do, however, provide a tool for making conservative comparisons of the load the drives might be required to sustain under alternative RAID configurations.

The RAID 5 equation component of (4*writes) implies that each database write incurs 4 physical I/Os, 2 reads and 2 writes; it is not meant to imply 4 physical write operations for each database write operation.

Application I/O Requests per Second =

RAID 0: (reads/transaction + writes/transaction) * transactions/second
RAID 1: (reads/transaction + 2*writes/transaction) * transactions/second
RAID 5: (reads/transaction + 4*writes/transaction) * transactions/second

The next step in the process is determining how many drives are required to support the applications I/O demands. Advertised I/O rates of today's typical 500 Megabytes through 2 Gigabytes drives can exceed 70 I/Os per second. However, these results are achieved with pure saturation testing without regard for request latency. Empirical testing has shown that these same disks sustain between 35 and 45 random database requests per second without incurring costly latency delays. Using this range you can determine the approximate number of physical drives to configure on the server and the drive size.

The following chart displays two lines which equate to 35 I/Os per second for the lower line and 45 I/Os per second for the upper line. Use this chart in conjunction with the previously discussed equations to determine the number of physical disks required by an application. Locate on the Y-axis the number of I/Os per second required by the application. Move across the graph and locate the intersection of the application I/O rate and the 35 to 45 I/O per second area. Determine the number of drives by locating the drive count on the horizontal axis.

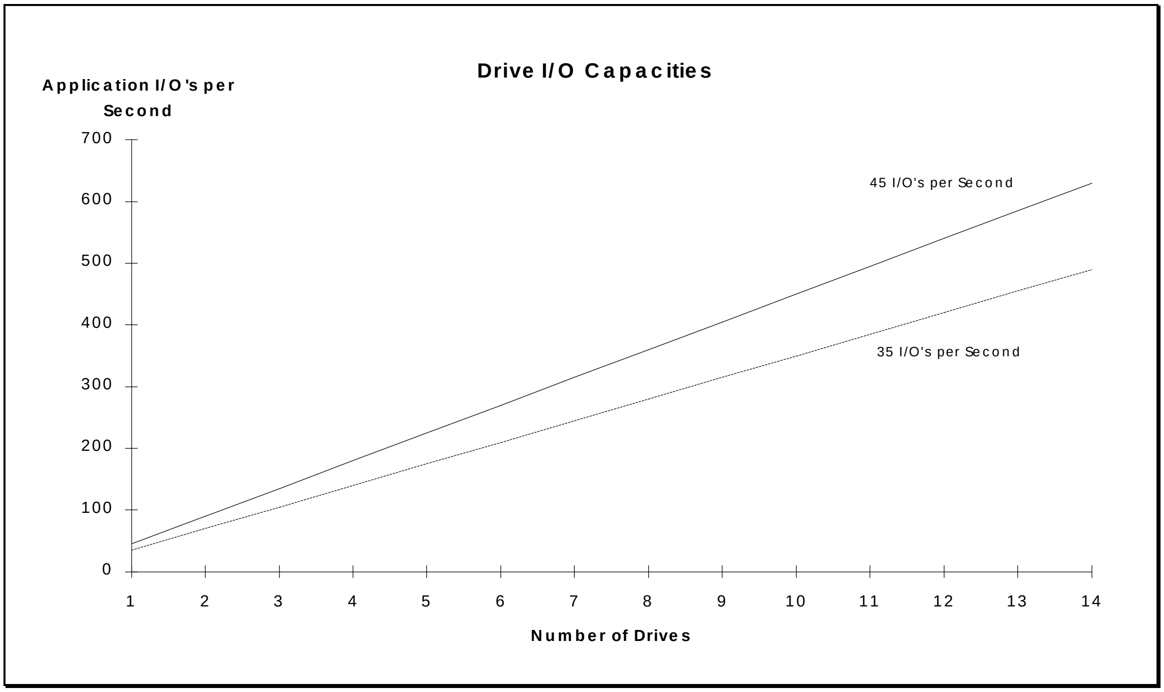

Figure 3-1. Drive I/O Capacities

The more conservative the implementation desired, the lower the I/O rates per drive that are required. If you are using RAID 1, you must have an even number of drives. If the result is an odd number of drives, round *up* to the next even number. When throughput requirements are more than the chart indicates, continue adding controllers and drives until you achieve the required throughput.

The total space requirements for the database and support files, combined with the throughput requirements and fault-tolerance, determine the size of disks which provide an optimal solution. When calculating total space requirements, add an additional 20 percent free space above the actual data requirements for growth and miscellaneous overhead. Again, a set of equations provide the answer to the appropriate size drive.

Drive size requirements based on total space needs and fault-tolerance:

RAID 0: [total space / number of drives] Rounded up to next drive size
RAID 1: [total space / (number of drives / 2)] Rounded up to next drive size
RAID 5: [total space / (number of drives - number of controller volumes)]
Rounded up to next drive size

Example Cases

are presented for each case. Some cases still require a judgment call by the implementor to choose the number of drives or drive sizes, but the alternatives are narrowed significantly. The alternatives chosen are conservative configurations and the reason for the choice, if not a single alternative, is explained.

Case 1: Complex transaction processing system requiring multiple queries per update.

Reads: 100/transaction
Writes: 5/transaction
Transaction rate: 2/second
Space Requirements: 3-Gigabyte Database + 20 percent Free space = 3.6 Gigabytes

RAID 0: (100 + 5) * 2 = 205 I/Os / Second
205 I/Os / Second => 5 or 6 Drives
3.6 / 6 = .6 => Round up to next drive size => 1-Gigabyte Drives

A conservative implementation yields six 1-Gigabyte drives for the data volume.

RAID 1: (100 + (2*5)) * 2 = 220 I/Os / Second
220 I/Os / Second => 6 or 8 Drives
3.6 / (8 / 2) = .9 => Round up to next drive size => 1-Gigabyte Drives

A conservative implementation yields eight 1-Gigabyte drives for the data volume. The alternative of using 6 drives was not chosen due to possibly lower performance and 2-Gigabyte drives would be required to meet space requirements. Using 2-Gigabyte drives yields a higher-cost system in and lower performance.

RAID 5: (100 + (4*5)) * 2 = 240 I/Os / Second
240 I/Os / Second => 6 or 7 Drives
3.6 / (7 - 1) = .6 => Round up to next drive size => 1-Gigabyte Drives

A conservative implementation yields seven 1-Gigabyte drives for the data volume.

Case 2: High-end OLTP system with several hundred users and a large database.

Reads: 20/transaction
Writes: 8/transaction
Transaction rate: 15/second
Space Requirements: 10-Gigabyte Database + 20 percent Free space = 12 Gigabytes

RAID 0: (20 + 8) * 15 = 420 I/Os / Second
420 I/Os / Second => 10 to 12 Drives
12 / 12 = 1.0 => Round up to next drive size => 1-Gigabyte Drives

Twelve 1-Gigabyte drives for the data volume was chosen for this case. If 10 drives were used, 2-Gigabyte drives would be required yielding a more expensive, lower performance system. However, the positive side of 2-Gigabyte drives would have been more available space for growth.

RAID 1: (20 + (2*8)) * 15 = 540 I/Os / Second
540 I/Os / Second => 12 to 14 Drives
12 / (14 / 2) = 1.7 => Round up to next drive size => 2-Gigabyte Drives

Fourteen 2-Gigabyte drives are required for the data volume.

RAID 5: (20 + (4*8)) * 15 = 780 I/Os / Second
780 I/Os / Second => 18 to 22 Drives
12 / (22 - 2) = .60 => Round up to next drive size => 1-Gigabyte Drives

For this system, two controllers with 11 drives each using RAID 5 would be recommended. Notice in this example the value '2' was subtracted from the drive count instead of 1. This is required when the drive count exceeds the capacity of a single volume or controller. Each volume is using n+1 parity. Therefore, you must subtract 1 for each volume to determine usable space.

Due to the very high I/O rates, RAID 1 and RAID 5 have similar costs. The reason is that the RAID 5 implementation requires many drive above the absolute space requirements to achieve acceptable performance levels. The RAID 1 solution, on the other hand, fits the space requirements more closely.

Case 3: Complex decision support with several simultaneous users.

Reads: 1000/transaction
Writes: 10/transaction (temporary space for sorting & grouping)
Peak Transaction rate: 3/minute or 0.05/second
Space Requirements: 5-Gigabyte Database + 20 percent Free space = 6 Gigabytes

RAID 0: (1000 + 10) * 0.05 = 50.5 I/Os / Second
50.5 I/Os / Second => 2 Drives
6 / 2 = 3 => Largest drive currently available => 2-Gigabyte Drives
Three 2-Gigabyte drives are required for the data volume due to space requirements.

RAID 1: (1000 + (2*10)) * 0.05 = 51 I/Os / Second
51 I/Os / Second => 2 Drives
6 / (2 / 2) = 6 => Largest drive currently available => 2-Gigabyte Drives
Six 2-Gigabyte drives are required for the data volume due to space requirements.

RAID 5: (1000 + (4*10)) * 0.05 = 52 I/Os / Second
52 I/Os / Second => 2 Drives
6 / (2 - 1) = 6 => Largest drive currently available => 2-Gigabyte Drives
Four 2-Gigabyte drives are required for the data volume due to space requirements.

The general conclusion from this discussion is that more, smaller disks are best for systems with large numbers of users, high I/O rates, and high concurrency. Fewer, larger disks work best for environments with a few users, high I/O rates, and low concurrency requirements.

KEY CONCEPTS SUMMARY

Configuring the drive subsystem is the most critical part of setting up a database server for optimal operation. The key concepts are:

- Distribute I/O requests over an appropriate number of drives.
- Carefully choose a fault-tolerance option.
- RAID technology provides the best choice for critical, high-performance database servers.

Finally, not all systems react the same due to differences among application profiles. The information provided in this document is based on five years of experience with array systems, both in Compaq labs and in customer situations. However, Compaq recommends that you test these concepts in your environment before implementing them on a production system.

Appendix A

MULTIVENDOR INTEGRATION FROM COMPAQ

To help you successfully integrate and optimize your network or multivendor system, Compaq has developed a library of systems integration TechNotes for the NetWare, Microsoft Windows NT, and SCO UNIX operating system environments. TechNotes provide you with important information on topics such as network performance management, server management, and operating system interconnectivity.

PRICING

- Individual Reprints of Compaq TechNotes: U.S. $45 (CND $50)*

*Plus shipping & handling

HOW TO ORDER

To order by PHONE (credit card orders only), call:

- 1-800-952-7689 (In U.S.: 7 a.m. - 7 p.m. C.S.T.)
- 1-800-263-5868 (In Canada: 24 hours)
- 1-317-364-7281 (From outside U.S. & Canada: 7 a.m. - 7 p.m. C.S.T.)

To order by MAIL or FAX, send complete order information to:

Worldwide: (except Canada & France)	Compaq Fulfillment Center P.O. Box 4100 Crawfordsville, Indiana 47933 FAX: 1-317-364-0787
Canada:	Compaq TechNotes Compaq Canada, Inc. 111 Granton Drive, Suite 101 Richmond Hill, ON L4B 1L5 FAX: 1-800-668-1835
France:	Compaq TechNotes Compaq Computer S.A.R.L. 5, Avenue de Norvege 91959 LES ULIS Cedex France FAX: 1-331-64-46-51-81

REPRINTS	PART NUMBER
Insight Management Agents	
Integrating Compaq Insight Management Agents	145806-001
NetWare	
Performance Management in a NetWare v3.1x Environment	133399-002
NetWare Tape Backup Management Using ARCserve from Compaq	137535-001
Tape Backup Using ARCserve v5.01 - Windows Edition from Compaq	137672-001
NetWare 4 Performance Management	137989-001
ORACLE Server Planning in a NetWare v3.11 Environment	145143-001
NetWare Server Management	145231-001
Compaq IDA-2 Configuration for NetWare	145536-001
Compaq ProSignia Configuration Guidelines for NetWare v3.x	145537-001
Compaq Answers to Technical Support Questions for NetWare Environments	145710-001
NetWare SFT III v3.11 Installation	145905-001
NetWare Migration from 2.x to 3.12	181069-001
Microsoft	
An Overview of Windows for Workgroups with Compaq Products	137527-001
3COM Network Upgrade Guide	145054-001
Implementing Lotus Notes on Compaq Servers with Microsoft LAN Manager	145201-001
Microsoft SQL Server and Novell NetWare Requester for OS/2	145376-001
Drive Subsystem Performance with Microsoft LAN Manager 2.1	145517-001

Configuration and Tuning of Microsoft SQL Server for Windows NT on Compaq Servers	184207-001

REPRINTS	**PART NUMBER**
UNIX	
SCO UNIX in an SNA Environment Using CLEO DataTalker U/X	145081-001
SCO UNIX in an SNA Environment Using Computone LYNX	145116-001
SCO UNIX Database Servers in Novell Networks	145402-001
Printing in SCO UNIX and NetWare Integrated Environments	145538-001
SCO UNIX Connectivity to SNA Environments Over X.25 Networks	145583-001
Disk Mirroring with VERITAS VxMirror	145656-001
SCO UNIX IP Routing Over X.25 Networks	145711-001
Performance Tuning for SCO UNIX on Compaq Systems	145730-001
Performance Tuning for ORACLE7 on Compaq Systems Running SCO UNIX	145942-001
SCO UNIX and ORACLE7 Database Server	195818-001
Technology Overview	
Configuring Compaq RAID Technology for Database Servers	184206-001

COMPAQ TECHNOTE ORDER FORM

Ship To:

Name: ____________________ Company: ____________________

Address: ____________________ Dept/Mail Code: ____________________

City: ____________________ State/Province: ____________________

Zip/Postal Code: ____________________ Country: ____________________

Phone: ____________________

Payment Method:

❑ Company Check ❑ Cashier's Check ❑ VISA ❑ MasterCard ❑ American Express

❑ Personal Check (Driver's License Number/State -- USA only): _________ ____________________

Card Number: ____________________ Expiration: ____________________

Card Holder Name (please print) ____________________

Card Holder Signature ____________________

Part Number	Title	Qty	Unit Price	Total Price
		Shipping/Handling*		
		Subtotal		
		Sales tax**		
		Total		

**For U.S orders, add applicable sales tax.

For Canadian orders, add 7% GST

***SHIPPING AND HANDLING**	**U. S.**	**Canada**	**Outside U.S. and Canada**
COMPAQ TechNote Reprint	$2.00	CDN $2.50	U.S. $15.00

For more information, call:

1-800-952-7689 (USA)
1-800-263-5868 (Canada)
1-317-364-7281 (Worldwide)