# Compaq Backup and Recovery for Microsoft SQL Server 6.X

**White Paper**

---

**Prepared By**

**Database Engineering**

**Compaq Computer Corporation**

**July 1997**

---

 Doc No 444A/0797

# NOTICE

The information in this publication is subject to change without notice.



This publication does not constitute an endorsement of the product or products that were tested. The configuration or configurations tested or described may or may not be the only available solution. This test is not a determination of product quality or correctness, nor does it ensure compliance with any federal, state or local requirements. Compaq does not warrant products other than its own strictly as stated in Compaq product warranties.

Product names mentioned herein may be trademarks and/or registered trademarks of their respective companies.



***Compaq Backup and Recovery for Microsoft SQL Server 6.x***

Second Edition (July 1997)
Document Number 444A/0797

**Compaq Computer Corporation**

## Table of Contents

## List of Charts



## List of Tables



## List of Figures



## Introduction

The purpose of this document is to provide the reader with information on backup and recovery technology available for and specific to the Compaq ProLiant family of servers and Microsoft SQL Server for Windows NT. It is our desire to deliver the best technical information possible on a specific topic in a timely manner and in a highly usable format. Any comments, ideas and suggestions are always encouraged and appreciated.

Data backup and recovery is one of the most important aspects of a database server, for it is the data that is the most valuable asset of your company. This document will present and analyze various backup and recovery solutions available for your Windows NT and SQL Server platform, from both the hardware and software perspective, and provide you with choices based on performance, cost, capacity and functionality. Based on the information presented, you will be able to make a more informed decision about the protection of your data.

When planning or implementing a comprehensive data protection system, fault tolerance often comes to mind. Fault tolerance, whether hardware-based or software-based, complements, not replaces, backup and recovery. While the primary purpose of backup procedures is to store data on an auxiliary storage medium for long term retention and catastrophic failure recovery[1], fault tolerance protects against loss of data due to a hardware failure. In fact, for any business critical systems, where data availability is crucial and any downtime due to a hardware failure is very costly, fault tolerance is a necessity. A detailed discussion of fault tolerance and related topics is, however, beyond the scope of this document.

Replication, on the other hand, is a form of fault tolerance at the server level. It involves two or more servers, which all contain a replica of the primary database, synchronized either real-time or near real-time[2]. A fail-over server solution can also be set up where in the event of the primary server failure, a designated secondary server mounts the storage and database of the primary server, and all user requests are then rerouted to this secondary server. A detailed discussion on replication or fail-over servers is however, beyond the scope of this document.[3]

The database administrator seeking a sound backup and recovery strategy today has more options available than ever before. Increased attention in the industry to the area of data archival has led to the emergence of powerful and rather versatile new backup products. We have attempted to make this document as comprehensive as possible to meet the needs of all Compaq customers who manage SQL Server environments on our hardware products. The customer who is beginning to plan such an environment may benefit from a comprehensive reading of the document in order to develop a greater understanding of his or her backup / recovery options. Those who are seeking to improve upon an existing backup / recovery strategy by leveraging new technology, may better be served by browsing the first two chapters (which stress concepts and considerations), then referencing those sections of the third chapter (which covers methodology and performance) which are of relevance or interest.

---

[1] Other uses of backup include protection against accidental or malicious data deletions or modifications, theft, viruses, etc.

[2] It should be noted that the native replication capability of SQL Server 6.x is not specifically designed for use as a ‘hot backup’ system.

[3] For more information on fail-over server solutions from Compaq, refer to the document *Configuring Compaq Recovery Server with Microsoft SQL Server 6.5.*

## Changes from Previous Version

The information contained in this documents updates and adds to the information contained in the paper *Compaq Backup and Recovery with Microsoft SQL Server,* which was released in Oct. 1994. The additions in this paper serve mainly to inform the reader on new backup related software and hardware technologies that have appeared on the market since then. These technologies include:

**Software:**

- SQL Server 6.0 and 6.5 from Microsoft.
- ARCserve 6.0 for Windows NT from Cheyenne.
- ARCserve Backup Agent (DBagent) for SQL Server from Cheyenne.
- ARCserve RAID Option (Tape Array Software) for Windows NT from Cheyenne.

**Hardware:**

- 10/20-GB, 15/30-GB, and 35/70-GB DLT Tape Drives from Compaq
- DLT Tape Arrays from Compaq
- Fast-Wide SCSI-2 Technology from Compaq
- SMART-2 Drive Array Controller from Compaq
- 100-Mbit networking products from Compaq

Details on each of the above technologies will be provided later in this paper. Of these technologies, the most significant for the purposes of this paper (backup and restore) have been the introduction of Microsoft SQL Server 6.x and the Compaq DLT tape drives. The previous paper, *Compaq Backup and Recovery with Microsoft SQL Server*, provided information pertaining to results achieved with SQL Server version 4.21a and the Compaq 4/16GB TurboDAT (and earlier) tape drives. Readers interested in these earlier technologies should refer to that paper. However, since some of the information from the previous paper continues to apply to the current environment parts of it have been retained in this document, including the entire first chapter entitled **Data Protection Concepts**.[4]

[4] Modifications have been made as needed.

## Data Protection Concepts

In the Data Protection Concepts section of this document, we'll characterize various data protection and recovery methods that are available to you in this environment. We'll touch on topics such as instance recovery and fault tolerance, but our main goal is to outline your options in the area of database backup and recovery.

### *Instance Recovery*

Automatic data protection and instance recovery consist of the system's ability to preserve the effects of all committed transactions and insure the database is in a consistent state after recovery from any single point of failure. This single point of failure includes a hardware component failure, software component failure, power loss, etc.

SQL Server provides this automatic data protection and instance recovery through its write-ahead log, called the ***transaction log***, where SQL Server records changes before it modifies the actual data pages. In the case of a server failure and recovery, SQL Server, upon startup, uses the transaction log to ***roll back*** uncommitted transactions, and ***roll forward*** transactions that were committed but not applied to the data files.

All data definition changes (such as creating or dropping tables, views, stored procedures, rules, etc.) and all data manipulation changes (inserts, deletes and updates, but not selects) are logged in the transaction log. Certain other events such as checkpoints and dumps are logged as well. Naturally, any data definition changes, checkpoints, dumps, etc., will not be rolled back or forward.

This data protection scheme cannot be turned off, and guarantees that data remains consistent at any point in time. The automatic data protection and instance recovery, in terms of the write-ahead transaction log, is inherent to SQL Server. More detailed information about transaction logging can be found in the Microsoft SQL Server manuals and other publications.

### *Fault Tolerance*

Fault tolerance and data archival should coexist on the same system. Fault tolerance, both hardware and software based, provides you with immediate protection against a hardware or a software component failure. In conjunction with hot-pluggable drive systems, hardware-based fault tolerance allows for continuous data availability, maximum performance[5] and automatic data recovery. Database archival allows for a long term data retention, and recovery from operational errors and catastrophic failures.

The drive subsystem has a substantially higher chance of failing than any other component in the system. Here, various levels of fault tolerance are available, ranging from no fault tolerance to fault tolerance for the entire drive subsystem. Naturally, higher levels of fault tolerance carry higher implementation costs. Many times, a high level of fault tolerance can seem to be a rather expensive solution. However, you need to weigh the cost of implementing a good, fault tolerant system against the cost associated with down-time. Sites requiring access to data with minimal down time must implement hardware fault tolerance for the entire drive subsystem.

With fault tolerance implemented on the entire drive subsystem, a drive failure will not interrupt any operations and will not cause any data loss. With the Compaq SMART and SMART-2 Array controllers and the hot-pluggable drive subsystem, a failed drive can be replaced and fault tolerance restored without taking the system off-line.

[5] For performance characteristics of various RAID levels of the Compaq SMART Array Controller, and configuration considerations, refer to the document *Configuring Compaq RAID Technology for Database Servers*.

With disk fault tolerance installed on the transaction log volume only, and with systematic database and transaction log dumps, your data is also safe. If you have a full database dump and subsequent transaction log dumps, you can easily restore your data in case of a failed data drive. This recovery involving the data drive will, however, require the system to be brought off-line. If you lose a transaction log drive, this volume is protected by fault tolerance, and recovery will consist of only replacing the failed drive and letting the controller rebuild the new drive to full fault tolerance.

With no fault tolerance installed, you run into a risk of losing transactions even in the case of frequent backups. Any changes to your database that occurred after your transaction log dump will be lost if your transaction log drive fails. In addition to a high possibility of data loss, recovery will require the system to be taken off-line.

More detailed information on the backup and recovery procedures in various failure scenarios, refer to the SQL Server Administrator's Guide.

## *Database Backup and Recovery*

Your ability to recover your data depends entirely on your data protection scheme. This protection scheme can consist of various methods, such as implementing database and transaction log backups alone, combining these backups with fault tolerance, or even taking advantage of replication. In this section, we'll provide you with some more insight into backup and recovery options available with Microsoft SQL Server for Windows NT, and refer you, whenever possible, to a more detailed source of information.

### Backup and Recovery Strategies

Database backup and recovery consists of the database administrator archiving data onto a secondary storage on a scheduled interval, and in the case of a failure, restoring the data from this secondary storage. Secondary storage can be a remote backup server, a local hard disk device, a local tape device or a local optical disk device.

Database archiving can be accomplished in two ways. The database administrator can shut down SQL Server, thus closing all database and transaction log files, and then perform a file-based backup of all SQL Server devices. This type of backup is called ***off-line backup***. It is necessary to stop SQL Server to gain access to the files, since SQL Server keeps all database and transaction log files open with exclusive access as long as the database engine is running. Therefore, any attempt to backup these files while SQL Server is running will fail.

Or, the database administrator can use SQL Server's backup facility to dump the contents of the transaction log, the database, or both, while SQL Server remains running. This type of backup is called ***dynamic backup*** or ***on-line backup***. SQL Server backup operation is referred to as a ***dump***, and the restore operation is referred to as a ***load***.

Choosing the correct backup and recovery strategy involves a careful analysis of your environment. Some business critical environments can accept only one strategy: on-line backup and recovery. Others can employ either one. Below we present some of the major characteristics of each strategy, and in the rest of the document we provide more detailed information, especially in the area of performance.[6]

### Off-Line Backup Characteristics

Shutting SQL Server down for archiving purposes has the following characteristics:

- ❑ Any and all user databases remain inaccessible for the duration of the backup operation. SQL Server opens all databases at startup and keeps them open whether they are being accessed or not. Therefore, you cannot use typical file backup utilities while SQL Server is running.

[6] This document will provide a performance analysis of online backups only.

- Any file backup utility will backup the files in their entirety. To illustrate our point, let's use an example where we allocate a 100MB database device, create a 100MB database on this device, and store 50MB of data in this database. From the operating system's perspective, we have 100MB worth of data, which we need to back up. However, we end up backing up about 50MB of data and 50MB of empty database structures, increasing both the duration of the operation and amount of the backup storage used or needed.
- You have to back up all database and transaction log files in order to recover. SQL Server keeps track of all user databases in the master database, and time stamps all files. If you restore only selected files, SQL Server will detect inconsistency between the files and the information in the master database. Likewise, in order to restore a database, you have to restore all database and transaction log files to keep them all consistent.
- Incremental backups (backups of changes since the last full backup) or partial backups (backups of selected subsets of data, such as a particular database) are not possible.
- Disk I/O associated with file-based backup utilities typically occurs in 64KB blocks, thus improving performance.
- Some file backup utilities support software data compression. Software data compression can yield the fastest backup throughputs, but is demanding of the system CPU.
- Off-line backups can take advantage of verifying the data (backup with verify) on the tape after backup. This guarantees data integrity on the tape.
- With Cheyenne ARCserve for Windows NT and Compaq TurboDAT AutoLoader or Compaq DLT Library, you can group tapes together to increase the capacity. Groups of tapes appear to the host as one continuous tape with an increased capacity, and the next tape in the group is automatically and transparently loaded once one tape is filled up.
- File-based backup utilities cannot be used to archive a database that has been created on raw devices. Creation of database device files on 'raw' (unformatted) partitions is supported by both Windows NT and SQL Server to enhance I/O performance by bypassing the file system. The only way to archive such a database is by using online backup.

### On-Line Backup Characteristics

SQL Server-based database and transaction log backups have the following characteristics:

- SQL Server will dump a database or a transaction log while permitting access to any and all user databases. All processing in databases not being dumped continues normally. Any changes[7] to databases not being dumped remain unaffected. In the database being dumped, any change to a data page already dumped occurs immediately. This change, however, will not be included in the backup image. If a transaction intends to update a data page or an index page not yet backed up, SQL Server places this transaction on an internal queue, dumps that page before allowing any change to this page, and then proceeds with the transaction. In other words, the database image is the data captured at the instant the DUMP command is issued. We will discuss performance implications of this type of backup later in the document.
- SQL Server will backup only pages containing data - both data and index pages. As a result, no unnecessary data will be backed up. However, during the load operation, SQL Server will initialize all unused pages in addition to reloading all used data and index pages, thus consuming more time as compared to the dump operation.

[7] We are only concerned with insert, delete or update requests. Look-up (read-only) queries don't concern us here since they have no affect on the user data.

- On-line archiving allows for incremental backups (transaction log dumps) and partial backups (dumps of selected databases).
- On-line dumps can take advantage of hardware data compression, if one is available via the tape drive. Software data compression is not, however, available.
- Since on-line dumps allow backups while the database is active, it would not be efficient to perform a verify operation after the backup. Therefore, the verify operation is not supported. SQL Server, however, reports any errors it encounters via the error log and the event log.
- The major limiting factor of previous versions of SQL Server-based backup was the performance, mainly due to the size of the I/O requests generated, both for the data (disk) volume as well as for the tape device. Improvements have been made to SQL Server 6.x which increase I/O throughput (see the later section entitled **Enhancements in SQL Server 6.x**), so that the performance of On-Line backup is not much below that of Off-Line methodologies. We will analyze the performance implications of SQL Server-based backups in greater detail later in this document.
- SQL Server currently does not support the autoloading and/or cataloging capabilities of the Compaq TurboDAT AutoLoader or DLT Library. However, third party software that supports these products along with a database 'backup agent' for SQL Server can be used.

The scope of this paper is limited to describing backup functionality available either directly from SQL Server or from a third party utility to be used in conjunction with SQL Server functionality. Thus, later sections of this document will cover the subject of **online backups only**, and information on offline backup software and methodologes will be addressed in other documents.

## Transaction Log vs. Database Archiving

Database dumps create complete images of both the database and the transaction log, take more time to perform as opposed to transaction log dumps, and consume more of the backup media. A recent database dump, however, may provide a faster data recovery, if one is needed, as opposed to loading an older database dump and a number of transaction log dumps created afterwards.

To illustrate this point, suppose that you create a database dump on Sunday, and each weekday you perform a transaction log dump. If you experience a failure on Saturday, you have to restore the database image created on Sunday and five transaction log dumps created during the week. However, if you create a database dump on Sunday and again on Wednesday, with transaction log dumps on the days in between, to recover from the failure on Saturday you would have to restore the database image created on Wednesday and follow it with restores of only two transaction log dumps, one from Thursday and one from Friday.

Transaction log dumps[8], or on-line incremental backups, create only images of the transaction log, take substantially less time to perform, and consume only a fraction of the backup media as compared to database dumps. A high number of transaction log dumps since the last database dump will prolong your data recovery, as illustrated in the example above.

The strategy involving these types of backup and their frequencies depends entirely upon your requirements. A careful analysis of your needs, along with techniques outlined in the System Administrator's Guide, will help you determine a plan that is best for you.

## Storage Destinations

You have many options for storing your backup images. You can back up your data to a local tape drive, a hard disk drive or a local rewrittable optical disk drive. Or you can back up to a remote server, which can store the data on its tape drive, hard disk drive or rewrittable optical disk drive. This section will explain the advantages and disadvantages of each type of storage destination.

[8] Transaction log dump requires the transaction log to reside on its own device file.

### Local Backup vs. Remote Backup

Local backup consists of archiving your data to a storage device physically attached to the database server. This device can include a tape drive, a hard disk drive or a rewritable optical disk drive. You can also use a separate disk drive for temporary storage, and move the image to a tape at some later time.

Remote backup consists of archiving your data to a storage device located on another server. This device can consist of a hard disk drive or a rewritable optical disk drive. SQL Server does not support direct dumps to a remote tape, even though you can dump your data to a remote server's shared disk drive and then move it to a tape local to that server. The remote server can then act as a database backup server, to which multiple database servers dump their data.

With a local backup implementation, each database server requires its own backup storage. This storage is typically a tape drive, even though it may also be a separate disk volume[9]. There is no additional network overhead generated, but users of this server will experience an increased response time due to the additional workload induced by the backup activity.

With a remote backup implementation, the cost of the backup server hardware and software can be shared among many database (and other types of) servers. You may, in some cases, even utilize an existing file server as your database backup server. Additional overhead generated by increased network activity can be alleviated by scheduling backups for after-hours, or by dedicating a separate, high-speed network link between the database server and the backup server.

### Tape Backup vs. Disk Backup

A SQL Server logical dump device can point to either a tape drive (e.g.: **\\.\Tape0**) or to a file on a disk partition (e.g.: **D:\Dump1.DAT**).

Backups done directly to a tape device are a convenient and relatively inexpensive way of backing up your database. The capacity and the speed are dependent upon the tape hardware. The tape device must be physically attached to the database server to perform SQL Server dumps to tape, unless special database backup agents are used[10]. File-based tape backup utilities support archiving of remote data, but the backup software must be running on the same machine as the tape drive.

Dumps done directly to a disk device, local or remote, are typically faster but more costly as compared to tape backups. You can consider dumping directly to a disk device and then moving the image onto a tape.

### <u>Backup Frequency, Scheduling and Retention</u>

Frequency of your database and transaction log dumps, scheduling of these dumps and the retention period all depend entirely on your environment. Factors such as an acceptable amount of work that could be lost, if any, acceptable down-time due to a recovery from a failure, the volume of update transactions, etc., will all influence the backup frequency, scheduling, and data retention. SQL Server offers a "media retention" option, configurable through *sp_configure*, which allows you to prevent overwriting an existing dump too soon. Automatic scheduling is available via the SQL Server Executive service or third party software. Refer to the System Administrator's Guide, documents available from Microsoft, or your third party software documentation for more detailed information.

---

[9] Using the same physical disk volume as your database or transaction log volume is not recommended for two reasons: 1) if your dump destination volume is the same as your database volume, and you lose a disk on this volume, you have no way of recovering your data, and 2) isolating I/O generated by the dump activity from the transaction log and database I/O activity minimizes performance impact.

[10] Database Backup Agents are available from Compaq or Cheyenne for use with Cheyenne ARCserve, and are covered later in this paper.

## Recovery Considerations

The recovery operation consists of restoring your data from the backup media. You will use the same utility to restore your data as you used to back it up - if you used a file based backup utility, you will use the same utility to restore your data, and if you used SQL Server to dump the database or the transaction log, you will use SQL Server to load your data. The same considerations that apply to backups also apply to restores.

Restoring your database from an off-line backup involves restoring all database and transaction log files, including your master database file. All SQL Server databases remain unavailable until all files are restored.

Restoring your database from an on-line backup typically involves dropping the damaged database and lost devices, and recreating both[11]. This creation of the database framework can be very time-consuming, depending on the size of your database, and adds to the total time needed to restore your data. Then, after all database devices are created and an empty database is built, the most recent database dump can be loaded, followed by loading all subsequent dumps of the transaction log. The entire restore procedure locks the database and disallows any use. Other undamaged databases remain on-line and accessible.

The level of success to recover all your data and the time required to do so depend on your backup strategy. The most important consideration is, if you can't restore your data, your backups are worthless. Therefore, spend time and effort to test your backup AND recovery strategy!

More details on various SQL Server-based recovery scenarios, load requirements and other considerations can be found in the System Administrator's Guide or the manuals available with your backup software. We will analyze some considerations and performance characteristics later in this document.

## Other Backup and Recovery Tidbits

- With SQL Server dumps to a disk device, each subsequent dump automatically adjusts the size of the existing file to reflect the amount of data dumped. For example, you initially dump 100MB to a DISKDUMP.DAT file, copy it onto tape, but do not delete the file. Then, the next day, you dump a transaction log (25MB of data) to the same dump device. SQL Server for Windows NT will adjust the size of the file to 25MB. Alternatively, you can have SQL Server append successive dumps to the same file (this is now the default in SQL Server 6.x).
- Microsoft recommends running database consistency checking on the database before performing database dumps. This checking includes *DBCC CHECKDB*, *DBCC CHECKALLOC* and *DBCC CHECKCATALOG*. If the database contains consistency errors, these errors will be included in the dump, and in the database once it is reloaded. These consistency errors may even prevent the database from being loaded. The execution of these consistency commands may, however, take extended periods of time. Refer to Microsoft SQL Server documentation for more information.
- Databases can be transferred between SQL Servers by doing an online dump at one server and then loading the dump into a database of the same size or larger at another server. However, when exchanging data this way, certain compatibility issues must be kept in mind: 1. Both SQL Servers must have the same Code Page and Sort Order. 2. Both servers must be of the same processor architecture (x86, MIPS, etc). 3. A dump done at a version 4.21x SQL Server can be loaded at a version 6.x SQL Server, but not vice versa. 4. A dump done at a version 6.0 SQL Server can be loaded at a version 6.5 SQL Server, but not vice versa. 5. A dump done at a Microsoft SQL Server cannot be loaded at a non-Microsoft SQL Server. If you encounter any of the above issues your

[11] If the database is not damaged, you can reload the backup over the old database.

restore will fail, and you will have to use a raw data import/export tool such as SQL Transfer Manager or BCP.[12]

[12] Even these utilities have certain limitations. See the Administrators Guide for details.

## Backup and Recovery Considerations

In this chapter, we will analyze some key benefits and limitations of various backup and recovery products, and provide some considerations to help you choose the hardware and software best suited for your environment.

### *Tape Products*

In this paper we have chosen to evaluate the Compaq DLT (Digital Linear Technology) family of tape drives, which includes the 10/20-GB DLT, 15/30-GB DLT, and 35/70-GB DLT tape drives. Other, older tape technologies such as the QIC (Quarter Inch Cartridge) and DAT (Digital Audio Tape) drives are also supported by Compaq but are not discussed here.[13]

#### **DLT Technology, Formats and Cartridges**

The Compaq Digital Linear Tape (DLT) drives were designed to meet the growing backup needs of large computing environments, providing greater performance, capacity, and reliability from a PC server tape solution than ever before. The DLT ensures data integrity through multiple checks including: error-correction-code (ECC) on every 64KB of tape data, cyclical-redundancy-check (CRC) on each 4KB of tape data, end-to-end CRC on each record overlapped with parity from the SCSI bus, and internal parity checking on the 2 MB memory buffer[14]. DLT drives record and read data using multiple, parallel tracks in a serpentine (alternating end-to-end) fashion. The DLT tape media supports a 128-track format, and DLT cartridges are specified to last 500,000 passes over the read/write head which equates to 7,812 backups using the entire tape capacity. The head life of the drive itself is estimated at 30,000 hours under continuous operation.

All Compaq DLT drives provide built-in hardware data compression using the DLZ (Digital Lempel-Ziv) algorithm. The 10/20-GB DLT drive (DLT2000) supports 3 density formats, the most efficient of which yields 10GB of storage space uncompressed and 20GB with 2:1 compression. The 15/30-GB DLT drive (DLT2000XT) supports 4 density formats, the most efficient of which yields 15GB of storage space uncompressed and 30GB with 2:1 compression. The 15/30 supports a superset of the 10/20 formats, so that it can read any tape recorded on a 10/20; but the 10/20 cannot read a tape that was recorded on the 15/30 using the highest (15GB) format. The 35/70-GB DLT drive (DLT7000) supports a superset of the other drives' formats, and offers a native format providing 35GB of capacity, or 70GB with 2:1 compression.

The 10/20-GB and 15/30-GB DLT drives do not differ in performance, and are both capable of 4.5GB/hr (1.25MB/sec) throughput with data compression disabled (up to 9.0 GB/hr with 2:1 compression). The 35/70-GB DLT drive however, offers not only enhanced capacity but also enhanced performance, being capable of up to 18.0 GB/hr (5.0 MB/sec) throughput with data compression disabled (36.0 GB/hr with 2:1 compression). These high-performance, high-capacity and high-duty-cycle backups make the DLT ideal for storage of large, mission critical databases like those found in many SQL Server environments.

Table 1 summarizes the tape cartridge support of tape drives evaluated in this document:

**Table 1 - Tape Cartridge Support Matrix**

| Tape Drive | Tape Cartridge(s) Supported |
|---|---|
| Compaq 10/20-GB DLT Drive | DLT Tape III (10.0-GB) |

[13] Results including Compaq QIC and DAT tape drives were covered in the previous paper, *Compaq Backup and Recovery with Microsoft SQL Server*, document #184489-001.

[14] The 35/70-GB DLT has an 8 MB on-board memory buffer.

| | |
|---|---|
| Compaq 15/30-GB DLT Drive | DLT Tape III (10.0-GB)<br>DLT Tape IIIXT (15.0-GB) |
| Compaq 35/70-GB DLT Drive | DLT Tape III (10.0-GB)<br>DLT Tape IIIXT (15.0-GB)<br>DLT Tape IV (35.0-GB) |

**Interface**

The Compaq 10/20 and 15/30 DLT drives are available with either an internal or external standard SCSI (8-bit) interface. These drives can however, be connected to a Fast-Wide SCSI-2 controller with the appropriate cable, although half the bandwidth of the Wide SCSI-2 bus will remain unused. On EISA systems such as the Proliant 4500 using a Fast SCSI-2 or a Fast-Wide SCSI-2/E controller will yield the same performance. For best performance on Pentium-Pro based PCI systems, Compaq recommends that the Compaq Wide-Ultra SCSI controller be used with these drives; the reasons for this pertain not to the enhanced bandwidth of the Wide-Ultra SCSI bus, but rather to PCI bus utilization issues which can prove significant in terms of system throughput - especially when using multiple such controllers[15]. The Fast-Wide SCSI-2 controller embedded on the Proliant system board however, can be used in conjunction with additional, Wide-Ultra controllers without degrading performance.

The Compaq 35/70 DLT drives are available with either an internal or external Wide SCSI-2 (16-bit) interface. Like the other DLT drives, the 35/70's should be used with the Compaq Wide-Ultra SCSI controllers on Pentium-Pro based Proliant systems if peak performance is desired.

Multiple drives can exist on the same SCSI bus by Daisy Chaining them with the appropriate cables. Compaq also provides DLT Tape Array cabinets which support up to four internal DLT tape drives. The cabinets have two SCSI channels so that the drives can be attached to two separate SCSI controllers (2 drives each); or the drives can be individually attached to up to four separate controllers using additional cables. For best performance, Compaq recommends not attaching more than two DLT drives per controller; the reasons for this will be discussed later in this paper. Finally, the DLT does not interface to the Compaq SMART SCSI-2 Array controllers, which are intended only for disk drive connections.[16]

**DLT Tape Library**

For unattended backup and restore functionality Compaq offers the DLT 15-Cartridge Library, which consists of one or two 15/30-GB DLT tape drive(s), along with up to 15 DLT Tape cartridges in 3 magazines for as much as 225 GB of capacity (more if compression is activated). The Library features a robot arm for automatic tape changing, as well as an integrated bar code reader to uniquely identify each cartridge and create a complete catalog of all archived data - in conjunction with supporting software. Software that supports the DLT Library includes backup products such as ARCserve 6.0 for Windows NT and its tape changer option; SQL Server cannot interface with the drive by itself.

**Device Drivers**

When operating in a Microsoft Windows NT 3.51 or 4.0 environment, the Compaq 10/20-GB and 15/30-GB DLT drives use the **Digital TZ86, TZ87, DLT2000** driver (DLTTAPE.SYS) that ships with

[15] The Wide-Ultra SCSI controller uses the 53C875 chipset, whereas the Fast-Wide SCSI-2/P controller uses the 53C825 chipset. The 53C875 makes better use of the PCI bus by transferring more data per bus ownership.

[16] For details on how to implement any of the interfacing techniques mentioned in this paragraph, please refer to the appropriate product documentation.

the standard Windows NT product[17]. This driver is **not** available on the Compaq Software Support Diskette for Windows NT (NT SSD). The Compaq 35/70-GB DLT drive uses a separate driver (DLT7000.SYS) that is available from Quantum corporation[18]. Some third party backup applications (such as ARCserve for Windows NT) provide their own driver for the DLT family of tape drives; when using one of these applications the Windows NT driver must be removed or disabled. If the DLT is interfaced to a 32-bit Compaq SCSI-2 controller such as the integrated controllers found on Proliant and ProSignia servers, make sure that the **Compaq 32-Bit SCSI-2 Controllers** driver (CPQ32FS2.SYS) is installed from the Compaq Windows NT SSD.

### *Data Compression*

Data compression is typically available via hardware (tape drive or tape drive controller) or software (offline backup application). Speed, effectiveness (compression ratio) and levels of support vary from one solution to another, and from one data set to another. Data compression will reduce the time and storage required, at the expense of adding some processing overhead.

Hardware data compression is available with the DLT tape drives when using 10.0, 15.0, 20.0, or 35.0 density formats. In this mode, data is always compressed when written to the tape cartridge; however, the drive will read both compressed and uncompressed tapes. You can choose not to use data compression by manually de-selecting the compression option on the drive, or using tape software that supports data compression control.

The nature of the data determines the effectiveness of data compression, and will directly impact backup performance. Highly compressible data, which we will characterize below, will yield substantially higher backup throughput rates than data which is not compressible. Data that compresses well includes:

- Unallocated space within the database, that has never been used. This includes space on a SQL Server device that has not been allocated to a database or a transaction log, or space in a database or in a transaction log that has not been allocated to an object. Such space, however, DOES NOT include pages that have been used previously but marked by SQL Server as “unused” or “free.” Even if your transaction log is empty, i.e. after a transaction log truncate or a dump, its pages may contain previous transaction log entries and are only **marked** as empty. Any file-based backup utility treats such marked pages as containing data. Naturally, this does not apply to on-line dumps, since SQL Server “knows” which pages contain valid data and which do not, and dumps them accordingly.
- Repeated patterns within the data set. The compression algorithms will detect repeated patterns in the database (regardless of data type), such as repeated customer names, cities or states, phone number prefixes, credit limits, etc., and compress them at high ratios. Not only database tables or indexes have repeated patterns. Transaction logs can also have highly repeatable patterns. For example, in cases where the transactions modify rows of data with the same or very similar columns, such as date and time stamps, or any repeated values, two or more transaction log entries will have highly compressible data.

Chart-1 illustrates the influence of data compression and type of data on backup performance. We obtained these results with off-line backup, using an off-line backup application and 4/16 DAT drives.[19] The “Unallocated Space” column represents a backup of an empty database. This database consists of

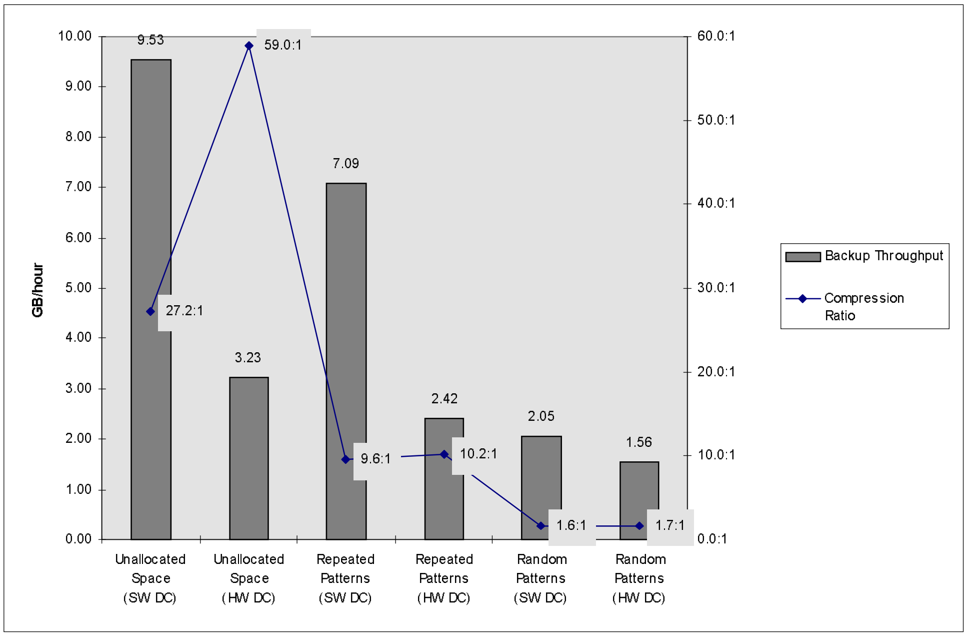
[17] Alternatively, the **Quantum DLT2XXX, DLT3XXX, or DLT4XXX tape drive** option may be selected from the Windows NT 4.0 product (this is the same driver - DLTTAPE.SYS).

[18] This driver should soon be available from Compaq as well.

[19] This data was incorporated from the earlier paper *‘Compaq Backup and Recovery with Microsoft SQL Server’ (1994)*. Although these tests did not use the same hardware and software that the rest of our document addresses, they still serve to illustrate the effects of compression.

initialized and never used pages only, and yields very high compression ratios and very high backup throughput.[20] The "Repeated Patterns" column represents a data set with a high percentage of repeated data and unused space, thus relatively high compression ratios and high performance. The "Random Patterns" column represents a data set with a very low to zero percentage of repeated data and unused space, thus low or no data compression. The chart should give you an idea how the compression ratios and backup performance vary, depending on the nature of data.

**Chart 1 - Data Compression Ratios and Performance**




From our results we can conclude the following:

1. Hardware data compression is typically more effective than software data compression, when both are operating on the same data set. This is an advantage for sites that want to conserve backup storage, and are not pressed to perform backups within the smallest time window possible.
2. Software data compression can sometimes be faster than hardware data compression, since it can utilize a more powerful and faster system processor (depends on the processor type, of course), and because it actually sends less data to the tape drive controller. Performance of software data compression is an advantage for sites that have a need to back up as fast as possible, with less concern over the backup storage consumed. Software compression by the application is only applicable to off-line backups, since it is not provided by SQL Server itself.
3. It is very important, with both hardware and software data compression, to keep the tape drive streaming[21] at all times to achieve optimal performance. If the system cannot keep the tape drive streaming, you may be able to achieve better performance without using any data compression. Here is why. Under no data compression, data is written to the tape "as is." Since no compression occurs, enough data is delivered to the tape drive to keep it continuously busy. With data compression, data is analyzed, compressed and then only a fraction of the original amount of data is actually being written. If the compressed data stream stops due to the host not keeping up, the

[20] The case of 'unallocated space' showing increased compression does not apply to a SQL Server based (online) backups.

[21] Streaming means that the system can and does deliver enough information to the tape drive to keep it constantly writing, as opposed to the tape drive having to stop and wait for new information.

tape drive writes out the current data, stops and waits for the data stream to start again. Once it does, the tape drive has to search for the end of data, reposition its write heads, and continue writing. This overhead is known as the head positioning overhead.

4. File-based backup software, with software data compression turned on, will typically utilize the CPU(s) heavily. To achieve optimal backup throughput in this case, make sure you have plenty of CPU power (a higher speed Pentium or Pentium Pro processor, or dual processors if the compression algorithm is multi-threaded). Use the Performance Monitor to determine your CPU utilization level. If your CPU becomes saturated, then your performance can suffer greatly. Hardware data compression, on the other hand, does not utilize the CPU(s) at all, and only the system I/O imposes some overhead on the processors.
5. When using the on-line backup method, your backup throughput can be limited by SQL Server and how fast it can perform reads from the disk subsystem. For SQL Server version 6.0 and above however, we will see that in most cases this is no longer a limitation.
6. With DLT Tape drives, the block size used to write the data to the tape media can become the limiting factor, so that the benefits of compression do not become apparent. We will discuss this further in the next section.
7. It is extremely difficult to predict the backup duration and throughput without knowledge of the compressibility of your data. Even once that is known, one can only predict, with a fair inaccuracy. In highly volatile environments, where the data changes rapidly, it is even difficult to use past experiences with the same database.

Consequently, for the rest of this document, we have chosen to present performance numbers with a database that has a very low percentage of repeated data and unused space, thus a low compression ratio[22]. Results with such a database should be treated as the “worst-case” scenario, and any database that contains compressible data would yield higher backup throughputs. Where applicable and known, we will provide additional considerations applicable for data sets with higher compression ratios.

## Estimating Hardware Compression Ratios

An advantage of software compression tools is that they usually give the user a number representing the amount of compression that was achieved on his/her data, expressed as a percentage or ratio. Some backup applications are able to give such a number for hardware compression as well (e.g: Cheyenne ARCserve). Without assistance from the application however, determining a precise compression ratio for your data using hardware compression can be difficult at best. In order to measure the compression achieved on our test database (see above) with the DLT tape drive, a special SCSI tape utility that counts blocks sent vs. blocks written was used. Short of such a utility however, it may be necessary to employ a bit of creativity.

One way to estimate compression is to back up a large, representative sample of your data to a single tape drive, and measure the time, or throughput[23]. Next, disable compression on the drive and repeat the operation. Divide the two numbers, and that is your estimated ratio of compression. For DLT drives, this method will only work with applications that employ a large block size, such as SQL Server 6.5 but not 6.0 (see following section). When using this method with SQL Server 6.5, we found that the number returned correlated quite closely to the compression ratio measured with the tape utility.

Another way to estimate compression is to back up your entire data to a single tape drive, while monitoring the progress. One way to do this is using the STATS = *percentage* option of the DUMP DATABASE command, which periodically returns the percentage of data pages dumped. Enough data should be present to fill up the tape, and the amount backed up should be recorded when the tape runs

[22]Our test database compressed approximately 1.13:1 (or ~11.75%) with software data compression, and approximately 1.18:1 (or ~15.25%) with hardware data compression.

[23] See the later section entitled **Calculating Database Dump Throughput.**

out of space. Next, disable compression on the drive and repeat the operation. Divide the two numbers to obtain a compression ratio or percentage estimate.

### *Block Size*

When data is written to the tape media, it is formatted for a certain *blocking factor*. The number of bytes per block is usually determined by the application when sending the data to the tape driver. When working with DLT Tape drives, this block size is an important consideration, as it can have serious performance implications. Ideally, the DLT drive can benefit from as large a block size as possible and the minimum block size recommended is 8KB. However, many applications do not use the larger block sizes due to in-memory buffering constraints, and some applications unfamiliar with DLT technology even use block sizes smaller than 8KB, because most other kinds of tape drives perform well with block sizes that can be much smaller (512 bytes or 1KB).

A well designed application will first query the tape driver with the appropriate API call to determine the kind of tape drive before attempting to set the block size. SQL Server 6.0 does exactly that, and sets the drive's block size to 8KB if a DLT drive is detected.[24] Actual SCSI data transfers are done to/from the DLT drive in 64KB sizes across the SCSI bus, but are further formatted for an 8KB blocking factor by the DLT before writing them to the tape media. The drive is set for ***fixed block mode***, in which it sends or receives a fixed amount of 'blocks' in each data transfer, in this case 8 blocks per transfer.

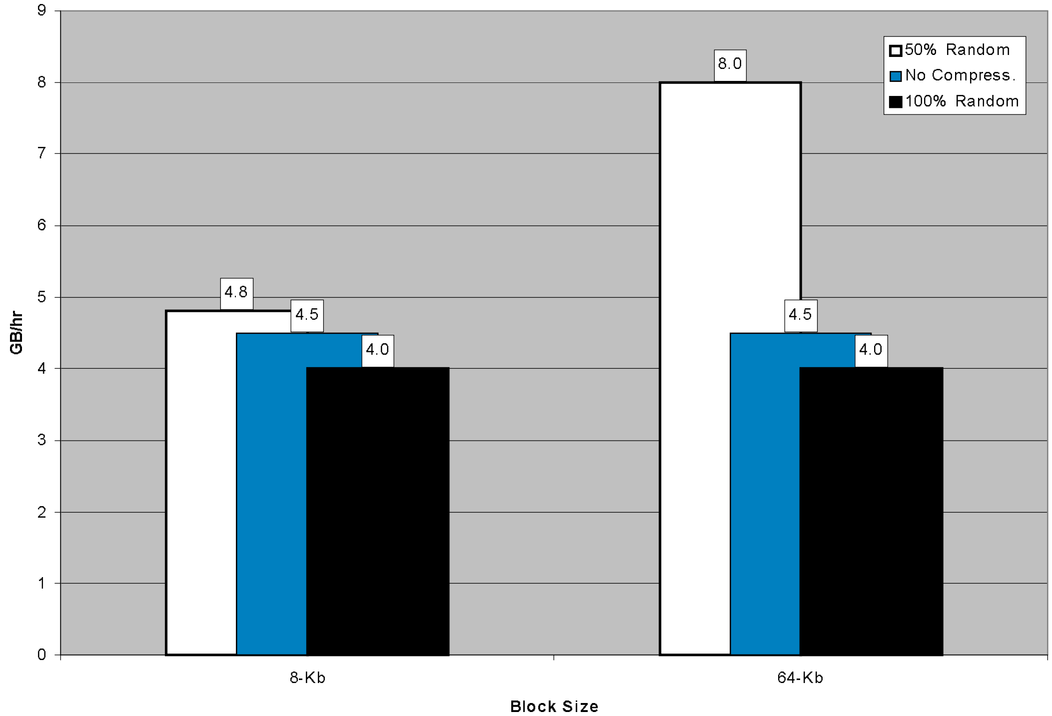
Even with an 8KB block size however, limitations will still be encountered. These limitations are most apparent when using compressible data sets. The overhead involved in formatting the data for an 8KB block size establishes a ceiling on how fast the DLT drive can write the data onto the tape. Thus the additional advantage gained by having the drive compress the data is not realized here. When an increase is made to a larger block however, the drive is able to write the data much faster so that the advantages of hardware compression become apparent.

The chart below contains data acquired by varying the block size used by a certain tape backup application under Windows NT, while also changing the patterns within the data file. For each block size, the bars on the left and right represent throughput achieved with data sets containing completely random patterns (should ideally yield 0% compression) and half random patterns (should ideally yield 50% compression), while the bar in the middle represents compression disabled on the drive. As can be seen, at 8KB it will generally make little difference what the compressablility of your data sets is, the results will not vary greatly beyond the standard rate of 4.5 GB/hr specified for the 10/20-GB and 15/30-GB DLT drives (this is the rate seen when compression is completely disabled on the drive). At 64KB however, the drive yields throughput more in line with the compression ratios of the data: with a 50% random data set we see an almost 2:1 gain in throughput. Also, note that when using a highly uncompressable data set, the throughput is slightly lower than when disabling the compression altogether, due to the overhead involved in having the compression logic process the data.

---

[24] SQL Server versions prior to 6.0 would use a hard-coded block size of 512 bytes, which had serious performance repercussions in the case of DLT tape drives. As a result, Compaq recommends that DLT drives not be used with these earlier (i.e: version 4.21x) SQL Servers. 512 byte blocks work fine for DAT and QIC drives.

**Chart 2 - Throughput Variance with Block Size & Data Patterns**




The above discussion is relevant to the subject of this paper in helping to explain performance differences that may be seen between versions of SQL Server. The blocking factor scheme used in SQL Server 6.0 was improved in SQL Server version 6.5 specifically for DLT drives. Whereas SQL Server 6.0 used an 8KB block size, version 6.5 now uses a much larger block size of 64KB.[25] Furthermore, 6.5 uses ***variable block mode***, in which only one block is sent to (or received from) the drive with each data transfer, but the size of the block can be varied during the dump (or load) operation. SQL Server 6.5 does not vary the block size though, so that the 64KB block is always used, thus providing for optimal performance with the DLT Tape drives.[26] The advantages of this scheme will become apparent in the later section entitled **Single Tape Device Dump.**

[25] SQL Server versions prior to 6.0 would use a hard-coded block size of 512 bytes, which had serious performance repercussions in the case of DLT tape drives. As a result, Compaq recommends that DLT drives not be used with these earlier (i.e: version 4.21x) SQL Servers. 512 byte blocks work fine for DAT and QIC drives.

[26] Exceptions are when writing headers, trailers, or file-marks to the tape, where the size of the data sent may be less.

## *On-Line Backup Considerations with SQL Server Dump*

This section will discuss the features, functional characteristics, and other considerations involved when using the native data "dump" facility provided with SQL Server. The database backup functionality built-into SQL Server 6.x is quite versatile. We will see that backup jobs can be run in a variety of ways, and that data can be archived to different types of storage. The performance benefits and drawbacks of each of these backup options can then be analyzed in subsequent sections.

### Enhancements in SQL Server 6.x

Microsoft has made significant enhancements to the backup functionality in SQL Server versions 6.0 & 6.5, which can yield better performance and ease of use than previous (4.2x) versions:

- **Striping Devices -** SQL Server 6.x now allows databases to be dumped to and loaded from multiple tape and/or disk devices, said to form a 'stripe set'. Unlike dumps or loads with single devices, striping is a multithreaded process and can significantly increase the throughput of the operation. Up to 32 backup devices can be used in a stripe set
- **Increased Concurrency -** Previous versions of SQL Server were not designed for more than 3 databases to be dumped simultaneously. With SQL Server 6.x up to 32 different databases can be dumped at the same time, provided that the corresponding number of backup devices are available.
- **I/O Requests -** Much of the performance limitations of backup and restore operations in the previous version of SQL Server were due to the sizes of the various I/O requests being done to and from both disk and tape. With SQL Server 6.x, all disk I/O (on the volume housing the database) during dump / load operations is now performed with requests for multiple data pages (up to 16KB per request), instead of just a single page (2KB size) at a time. The size of the data blocks sent to various dump devices has also been enhanced. When dumping to a disk device, large writes of 64KB size are performed, and when loading from a disk device 64KB reads are done. When dumping to or loading from a DLT tape device, 64KB data transfers take place. These larger I/O sizes allow SQL Server 6.x to more efficiently drive powerful hardware storage technologies such as disk arrays, Wide SCSI-2 bus, and Digital Linear tape.
- **Parameters -** SQL Server 4.2x used hard-coded values to determine the number of buffers and threads allocated when dumping to multiple devices, such as for parallel dump operations. Now, SQL Server 6.x has two new configurations parameters, ***backup_buffer_size*** and ***backup_threads***, used to control the size of the buffers and the number of threads used when doing parallel dumps or dumping to striped disk devices.
- **Append Dumps to Disk Files -** SQL Server has always provided the functionality to append new dump images to the end of existing ones on tape devices. With SQL 6.x, this functionality has been extended so that dumps can be appended to disk devices as well. Furthermore, the size of the file on disk will now reflect the size of the actual data image(s), as a dump file containing existing dumps is first truncated if the existing dumps are being overwritten, then extended as subsequent dumps are appended.
- **Single Table Operations -** With SQL Server 6.x, a single table within a database can be dumped to or loaded from a dump file. However, any table loaded in such a manner will reflect its state at the time the dump was made, and so may not be logically consistent with the current state of the other objects in the database (referential integrity is not guaranteed for single table loads). Single table operations are intended primarily as a disaster recovery mechanism (e.g: recovering a table that exhibits data corruption), and will not be covered further in this paper.
- **Dump to Named Pipe -** SQL Server 6.x now allows a named pipe to be specified as the dump device. This is useful when software vendors or customers want to write their own application to accept data from a SQL Server dump process either locally or across the network. In addition to named pipe dumps, SQL Server 6.x continues to support dumps to network disk devices, as did previous versions.

## Backup Characteristics of SQL Server

When considering an on-line backup, be aware of the following:

- SQL Server reads all pages from the disk devices, not from data cache. As a result, a bigger SQL Server data cache will not improve the performance of database dumps.
- Each dump or load is a single threaded operation, with the exception of striped dumps or loads in which one thread is used per device.
- A dump operation essentially requires the data pages from the database being dumped to be read directly from disk in a sequential nature. If you are dumping one database at a time to a single device, the speed of your backup operation will depend primarily on how fast your disk subsystem can perform single-threaded sequential reads. Dumping to striped devices adds additional threads to the process. SQL Server 6.0 can vary the size of the reads from a single 2KB page at a time to 8 pages (16KB) at a time[27]. The sequential nature of reads from the database during dumps can also benefit from read-ahead at the hardware level, such as the Compaq SMART-2 Array Controller which pre-fetches data into its on-board cache module.
- During a dump, SQL Server reads the data pages into in-memory buffers; with one buffer being allocated per backup thread. The size of these buffers can be tuned in 32 page increments from between 64KB to 640KB, using the ***backup_buffer_size*** parameter.[28] As each buffer fills up, SQL Server sends the data to the dump device using 64KB writes. In the case of tape devices, these writes are further formatted into block sizes suited for the type of tape media being written to. SQL Server 6.x determines the appropriate blocking factor by first querying the tape device using the appropriate Windows NT Tape API call[29]. In the case of DLT tape drives, SQL Server 6.0 uses a block size of 8KB (so that each write to the tape drive will contain 8 blocks of tape data), and SQL Server 6.5 uses a more optimal 64KB block size (1 block per write)[30].
- The database being backed up, as well as other databases, remains open for use. Naturally, the backup activity and queries against the database server conflict with each other, resulting in a performance degradation of both during the entire backup operation. This subject will be discussed in greater detail later in the document.
- A full database dump includes a dump of the log file, which follows the actual dump of the data. Like the data, the transaction log is dumped sequentially from beginning to end, and is read from the log device volume in sequential, 16KB size requests. The transaction log will not be truncated following a full database dump.

## Monitoring Transaction Log Usage

Preventing the transaction log from filling up is crucial for continuous SQL Server operation. The transaction log will fill up when it has not been dumped for a period of time and enough transaction log entries accumulate to consume all free pages in the log segment. Once the transaction log fills up, the corresponding database becomes unavailable for any insert, delete or update operations, or any other activity that generates a transaction log entry (dumps, checkpoints, object creations, etc.).

---

[27] During the testing for this paper, the average read size was measured closer to 16KB.

[28] The ***backup_buffer_size*** parameter is supposedly intended to tune dump and load performance. During our testing for this paper however, we saw that increasing this parameter from its default value of 1 either did not effect performance, or *negatively* affected performance by up to 50% (in the case of striped dumps to multiple DLT tape drives). We therefore strongly recommend that it be kept at its default value.

[29] SQL Server versions prior to 6.0 would use a hard-coded block size of 512 bytes, which had serious performance repercussions in the case of DLT tape drives.

[30] See the earlier section entitled **Block Size** for more information on this setting.

When the transaction log fills up and SQL Server is unable to record a checkpoint in the transaction log, you have to perform a dump of the transaction log, without making a backup copy of it, using *DUMP TRAN <database> WITH NO_LOG*. In such a situation, make sure to follow with a full database backup. Refer to the System Administrator's Guide for more details and important considerations regarding use of this command.

There is a number of ways you can monitor the transaction log usage and prevent the transaction log from filling up. Three methods are described below.

- Monitoring Transaction Log Usage with DBCC

  You can issue the *DBCC checktable(syslogs)* command[31]. SQL Server will update appropriate system tables and report the following:

  - total number of data pages in the *syslogs* table
  - total number of rows in the *syslogs* table
  - space used on the log segment in MB and in percent
  - space free on the log segment in MB and in percent

The *DBCC sqlperf(logspace)* command can also be issued, and will retrieve the transaction log space used for all databases that have the log on a separate device(s).

- Monitoring Transaction Log Usage with Performance Monitor

  In Performance Monitor, you can select *Object: SQLServer-Log, Counter: Log Space Used(%)* and *Instance: < database(s)>* to monitor the transaction log usage. Using the Alert option of Performance Monitor, you can generate administrative alerts triggered on a pre-defined transaction log usage, and have the administrator dump the transaction log upon reception of this alert. Or you can even automate this procedure and have a batch file dump the contents of the log.

- Truncating Transaction Log on Checkpoint

  You can issue the *sp_dboption <database>, 'trunc. log', true* command to have the transaction log automatically truncated at every occurrence of a checkpoint. Use this option **only** if, after a catastrophic data loss, you don't need to recover transactions which followed the last full database backup.

- Automating Transaction Log Dumps / Truncations with SQL Executive

  The SQL Executive Service in SQL Server 6.x includes several 'Managers' that can be used to automate transaction log management:

  - the Task Manager can be set up to perform dumps of the transaction log (or even the entire database) on a regularly scheduled basis (hourly, daily, weekly, or monthly). It offers the benefit of maintaining a task history, which can be used as a 'backup log'. This strategy assumes you know how often the dumps need to occur.
  - the Alert Manager can be set up to respond to a SQL Server event (a notice logged to the Event Log by the Event Manager), such as an event caused by SQL Error 1105 indicating that the Transaction Log is full. The Alert can then fire off a task to send an e-mail to the Administrator or to try and dump or truncate the transaction log. This is more of a 'reactive' strategy in that the transaction log is already full by the time the alert activates, and may

[31] The time to complete the *DBCC checktable(syslogs)* command depends on the size of your transaction log and space used; may take a long time to run on large log spaces.

require that a DUMP TRANSACTION WITH NO_LOG be performed followed by a full database dump[32]

- the Event Manager can be asked to log a SQL Server event in response to a Performance Monitor Alert. As described above, the Performance Monitor can be used to monitor log space usage and generate an alert. This performance monitor alert can either run a batch file to dump the log itself, or use the **xp_logevent** extended stored procedure to log a SQL Server event, which in turn can invoke a SQL Server alert that fires off its own task to dump the log. Either way, using Performance Monitor provides a more 'proactive' strategy to transaction log management while not requiring that you know beforehand how often the log dumps need to occur.

### Estimating Transaction Log Dump Size

The transaction log dump size can be estimated using the SQL Enterprise Manager utility, using the *Manage*, *Databases,* and *Edit Database* sequence of options. The output will include total log space and log space available in MB (subtract to get log dump size).

You can also use the *DBCC checktable(syslogs)* command. The output of this command includes the total number of data pages in the *syslogs* table and the space used on the log segment in megabytes. Use the number of megabytes used on the log segment, i.e. the number of data pages in the *syslogs* table multiplied by the page size of 2048 bytes, to approximate the size of the transaction log dump. Always allow for an extra 5% of the result to compensate for inaccuracy of the estimate. Alternatively, you can use the *DBCC sqlperf(logspace)* command. Also, the *DBCC updateusage* command can be run beforehand in order to correct possible inaccuracies in space usage reports.

### Estimating Database Dump Size

The database dump size can be estimated with the SQL Enterprise Manager utility, using the *Manage, Databases,* and *Edit Database* sequence of options. Enterprise Manager will give the total data space and estimate the database space available in MB (subtract to give dump size). The *sp_spaceused* stored procedure can also be used to report reserved space, data space used, index space used, and unused reserve space, all in KB. Add data space used and index space used to the log space used (see above).

The *dump database* command causes SQL Server to include a dump of the transaction log, to recover transactions that were in progress at the time of the dump. Always include the size of the transaction log in your estimate. Allow for an extra 5% of the result to compensate for inaccuracy of the estimate. Also, the *DBCC updateusage* command can be run beforehand in order to correct possible inaccuracies in space usage reports.

### Calculating Database Dump Throughput

Although the methods described above can provide a good estimate of the dump size, they tend to report 'rounded' numbers. The most accurate method to get the size of the dump image is to actually perform a dump to disk or tape, then to get the size of the file from a command prompt (not from File Manager) or from the SQL Enterprise Manager Backup/Restore window. This is the method used to measure the size of the test database used in later sections of this paper, in order to calculate backup throughput. Another, also accurate method of determining the dump image size is to read the number

[32] Depending on how full the transaction log has become, SQL Server may not be able to dump the log or even to do a normal truncate only. In this case a 'non-logged truncation' must be done which requires a subsequent database dump to preserve the usability of future transaction log dumps. For more information on this subject, refer to the *SQL Server Administrator's Companion.*

of pages reported as dumped in the SQL Server error log (or Windows NT Application event log)[33], then multiply this number by '2048'.

The time required to perform the dump can most accurately be acquired from the SQL Server error log (or Windows NT Application event log) as well. For dumps to tape, SQL Server will report the time that the tape is actually mounted for the dump operation in hh:mm:ss format. The tape mount time is logged after any initial period that may be required to rewind or search the tape. The time that the dump operation completes will also be reported in hh:mm:ss. The dump completion time is logged before any final period that may be needed to unload the tape. The tape mount time is not logged for disk devices however, but since dumps to disk are almost instantaneous the starting time can be gotten by running the *getdate* function in batch preceding the *dump* statement.

Dividing the dump image size by the time required for the dump operation will yield the throughput in MB/sec. This number can be multiplied by 3.6 to see the result in GB/hr.

Remember that a full database dump includes a dump of the transaction log as well. For the database dump performance testing done in the following sections of this paper, we first completely truncated the transaction log so that only actual data throughput would be reported. The throughput for the transaction log part of the dump may differ from the data part depending on where the log resides and how compressable it is. Transaction log dump performance is tested in a separate section of this paper.

### Database Layout Considerations

There is much that can be said on the subject of data layout as it pertains to database performance. Here we will consider only how the performance of a dump operation under SQL Server can be affected by the placement of your database across the disk drives. For a more comprehensive discussion of managing your database drive subsystem, please see the *Microsoft SQL Server Administrator's Companion*.

The most important thing to remember about the dump process is that it is completely sequential in nature: the process will begin reading data pages near the start of the database and continue in order until the end. Unlike with random I/O, if different sections of the database are located on separate disk volumes, this sequential process will be reading data from only one volume at any given time; and the overall backup performance may become limited by the read rate. The only way to guarantee that the reads will be issued to multiple drives simultaneously is for the database to be placed on a *single logical volume consisting of an array of drives*[34]. The best way to implement such an array is through use of a hardware RAID controller such as the Compaq Smart-2 SCSI Array controller.

For the purposes of optimizing backup throughput, when using the Smart-2 array controller it is best to create a single large array spanning the two SCSI channels, rather than to create multiple arrays and extend the database across the logical volumes on the separate arrays. A notable exception to this rule is when you have multiple large databases that you want to back up concurrently, in which case each database is best kept on its own array. The performance of data reads from various Smart-2 arrays as well as concurrent backup performance is covered in the subsequent chapter on **Backup and Recovery Performance**.

While a single Smart-2 controller will often provide enough capacity and throughput for a single database, some large databases may need to be built across multiple controllers. In this case, a number of methods are available to the database administrator for spanning two or more controllers:

- ❑ The database can be created on the first controller then extended onto the second controller, by using an ALTER DATABASE statement.

[33] The size (and time) of the dump upon completion may not be logged if more than 7 dump devices are used.

[34] Multiple logical volumes on the same array will achieve this result also.

- A SQL Server data device (.DAT file) can be created on each controller, then the database built so that portions of it alternate across each of the devices. For example, a CREATE DATABASE statement for a 40 GB database can be structured to create: 5 GB on dev1, 5 GB on dev2, 5 GB on dev1 again, etc. (8 sections total); where dev1 and dev2 are on separate controllers (there is a limit of 32 such 'sections' for any one database). This method has the benefit of 'spreading the data out' amongst the controllers more effectively than does extending a previously created database.
- A SQL Server data device (.DAT file) can be created on each controller, and one or more segments can be created per device or across the devices. This method requires that the data layout be done out at the object (table, index, etc.) level.
- A Windows NT stripe set can be created across the controllers. The stripe set effectively 'combines' the arrays on the two controllers into a single volume, and distributes data evenly across all of the drives.

With the exception of the Windows NT stripe set, the above methods will not cause an increase in read performance during a database dump, due to the sequential nature of the process (the process will read from only one controller at a time). Finally, remember that database layout affects more than just backup performance. While the hints in this section should be kept in mind when planning your database, the method and structure of the database layout can ultimately depend upon a number of things including the performance of the system under its normal production use.

## *Online Backup Considerations with ARCserve for Windows NT*

This section covers the use of a third party software program in conjunction with Microsoft SQL Server for the purpose of implementing database backup. As the SQL Server product matures and its use in the industry increases, many software vendors have designed utilities for use with SQL Server and some existing applications have been enhanced to become "SQL Server aware". In the area of backup and recovery, one widely used product that now has the capability to back up SQL Server databases online is ARCserve for Windows NT from Cheyenne Software[35]. The ARCserve family of products is available directly from Compaq to help form a storage management solution, and has been thoroughly tested on Compaq hardware. Thus, this section will focus specifically on ARCserve 6.0 for Windows NT, Enterprise Edition, and its 'add-on' option software. Subsequent chapters will then analyze the performance aspects of using the ARCserve products to carry out online backup of SQL Server databases.

### Features of ARCserve 6.0 and Options

ARCserve 6.0 Enterprise is a powerful storage management product that offers a rich set of features, especially when combined with its option software. Some of these features serve to provide ARCserve with capabilities beyond what is available with the native SQL Server backup, and so can be considered as advantages of using ARCserve over a straight SQL Server 'dump'. It is of course, up to the customer to evaluate both backup methods and to determine which type of backup solution best meets their needs. The following is a brief look at some of the capabilities of the ARCserve products:

- **Database Agent -** The Backup Agent for Microsoft SQL Server (formerly known as DBagent) is available as an option, and must be installed on the same system where the SQL Server is located in order to perform online backups there. Otherwise, ARCserve must perform off-line backups of the database device files. The agent functions as a Windows NT service, and allows backup at the database or table level.
- **Tape Spanning -** ARCserve 6.0 allows the creation of 'groups' of media, should multiple tape drives be available. A single backup job can continue uninterrupted across all the tape drives in that group. ARCserve supports up to 8 tape drives per server.

---

[35] Cheyenne Software, Inc. is now a wholly owned subsidiary of Computer Associates International, Inc.

- **Fault Tolerance / Striping -** This option provides a Tape RAID (or 'RAIT') system so that multiple tape drives can be placed in a group and be written to as one 'tape array' by a single backup job. Fault tolerance can be provided to the array through RAID-1(mirroring)[36] or RAID-5 (striping with parity), so that backup operation can continue if a tape drive fails, or so that a backup set can be recreated if a tape is lost. The array can also utilize RAID-0 striping, for a purely high-performance backup without fault tolerance. While the RAID-0 scheme is similar to the device striping offered by SQL Server 6.x, the fault tolerance capabilities of RAID levels 1 & 5 are beyond what is currently available with the native SQL Server product.
- **Auto Changer -** This option is for use with a tape autoloader or tape library such as the DLT 15-Cartridge Library from Compaq. The current version of the auto-changer is not compatible with the Raid Option however, so both cannot be used concurrently.
- **Parallel Streaming -** If tape drives are separated into different 'groups', then multiple backup jobs can be run (one to each group) simultaneously. This is similar to what can be done with SQL Server by dumping multiple databases concurrently. Parallel streaming also works with RAID tape groups.
- **Network Backup -** ARCserve allows for the online backup of databases on either the local server or on remote servers which have the SQL Server Database Agent installed. This way, the storage device(s) can be located on a single server, and data from multiple servers can be sent to this 'backup' server. Without ARCserve, SQL Server cannot do a database dump across the network to tape drive(s) located on another server, although it can use network disk dump devices. Conversely, ARCserve 6.0 cannot send data across the network to a disk backup device.
- **Centralized, Remote Management -** The Enterprise version allows the management of multiple servers from a single machine, using a 'tree' view display. All storage management tasks for these servers, including submission of backup and restore jobs, can be performed from a single location. This capability can be likened to the centralized database management capability offered through SQL Server's native Enterprise Administrator tool. The advantage with ARCserve is that the same console (i.e: the ARCserve Manager interface), can be used comprehensively for *all* backup related tasks on the network, including other file-based backup jobs.
- **Automation -** The ARCserve Job Engine makes it possible to schedule backup / restore jobs based on customized repeat methods (time interval, days of the week, etc.). In addition, a Tape Rotation scheme can be configured for the backup jobs. The SQL Server 6.x Task Manager, part of the SQL Executive service, provides a more limited scheduling scheme for native SQL Server backup tasks[37].
- **Information Repository -** The ARCserve Database Engine maintains complete historical information on such things as jobs that have been completed, files/directories/drives/machines that have been backed up, and the media that was used.

### Backup Characteristics of ARCserve

When considering an on-line backup, be aware of the following:

- The ARCserve Backup Agent 2.0 for SQL Server 6.x makes use of SQL Server's built-in backup capability. Using DB-Library functions, the agent initiates a 'Dump Database/Transaction' command within the SQL Server application. As such, most of the functionality of the 'dump'

[36] RAID-1 capability is not functional in the version 1.0 of the ARCserve RAID option, but will be in the next release.

[37] See the earlier section entitled **Monitoring Transaction Log Usage.**

process described in the earlier sections[38] applies when doing online backup with ARCserve as well.

- The dump process initiated in SQL Server by the Backup Agent always performs a *striped* dump using 4 devices (default), and thus invoking 4 read threads for increased performance, regardless of the number of tape drives actually being used for the backup job[39]. The dump devices specified are actually named pipe devices, for example **PIPE = '\\.\pipe\TID166\dbagent0s0'**. Thus as each SQL Server dump thread reads data pages from disk (typically in 16KB chunks), it sends the data to a corresponding pipe created by the Backup Agent. Like SQL Server, the Backup Agent is a multithreaded process, and dedicates one thread to each pipe in order to retrieve data in a parallel fashion. The agent sends the data to the ARCserve job process running on either the local server or a remote server across the network, where it is written to tape.
- The number of SQL Server pipe "devices" requested by the Backup Agent is tunable from within the Windows NT registry (up to 16). In HKEY_LOCAL_MACHINE, set '**StripeNum**' under:

**SOFTWARE->Cheyenne->DSAgent->CurrentVersion->agent->dbasql60**

For this change to work, the **backup_threads** parameter in SQL Server must be set correspondingly. In the testing done for this paper, we did not observe a significant difference in throughput by increasing this registry value.

- The default block size used by ARCserve to write data to DLT tape drives is normally 16KB. When the RAID Option is added to ARCserve however, the block size is increased to 64KB for performance reasons. This block size parameter is tunable from within the Windows NT registry, but we recommend leaving it at 64KB.
- The RAID Option will 'stripe' the data across the tape drives in an array, sending a 64KB block in turn to each drive. RAID-5 introduces a 'parity' block which alternates between each of the drives for every 'pass' across the array. RAID-1 simply creates an identical image on a second tape drive.
- ARCserve provides its own driver for the DLT family of tape drives. If the DLT driver supplied with Windows NT was installed, it must be removed or disabled so as not to interfere with ARCserve operation. Note that SQL Server dump operations directly to tape will no longer work once the Windows NT supplied driver becomes unavailable.

For more information on ARCserve 6.0 or its Option software, please refer to the User Guide for the corresponding product.

## Tape Capacity Planning

The ability to stripe data across multiple tape drives, such as that provided with the ARCserve RAID Option, offers us two key benefits: **1.** Increase in performance, and **2.** Increase in storage capacity. The performance aspects of tape striping will be considered in later sections. Here, we look at capacity. Tape striping can be used as a way to perform unattended backups, as long as capacity requirements are met. Normally, one would use a tape library such as the Compaq DLT 15-Cartridge library for unattended backups. However, if increased performance or fault tolerance is needed along with increased capacity, then tape RAID may be the better solution (currently, the ARCserve RAID option cannot be used to mirror or stripe data across both drives in the library, or across drives in more than one library).

A single 10/20-GB, 15/30-GB, and 35/70-GB DLT tape drives will provide a minimum of 10 GB, 15 GB, and 35 GB of storage, respectively. Therefore, 'n' such drives will provide 10n, 15n, or 35n

[38] See the earlier sections entitled **On-Line Backup Characteristics** and **Backup Characteristics of SQL Server**.

[39] The **backup_threads** parameter under SQL Server must be set to at least '3' when using ARCserve.

GigaBytes of storage in an ARCserve RAID-0 tape array. This applies to SQL Server tape stripe sets as well, which perform data striping similar to ARCserve RAID-0 arrays. ARCserve RAID-5 arrays will provide the capacity of n-1 drives, and at least 3 drives must be in the array. If the approximate compression ratio of your data is known, a more accurate estimate of the storage can be calculated:

- For ARCserve RAID-0 or SQL Server arrays: **Capacity = CR * d * #drives**
- For ARCserve RAID-5 arrays: **Capacity = CR * d * [#drives - 1]**

Where **CR** = Compression Ratio, **d** = density (10 or 15 or 35) depending on the type of drive, and the capacity is given in GB.

<u>Example</u>: Let's say a DBA has three databases totaling 65 GB in size which he wants to backup unattended overnight to a fault tolerant tape set. He must use an ARCserve RAID-5 tape array. If he knows the compression ratio of his data on DLT drives is roughly 1.25:1, then the number of 10/20-GB DLT drives needed can be figured as:

#drives = (65 + 12.5) / 12.5 = 6.2; which rounded up means 7 drives. If 15/30-GB DLT's were used, 5 drives would be needed, and if 35/70-GB DLT's were used then only 3 drives would be needed.

Earlier, we recommended that no more than 2 DLT drives be placed on a single SCSI controller for performance reasons, and this will continue to be stressed. However, if your goal is to increase capacity and not necessarily performance, and your system is constrained by the number of available PCI slots, then this rule can be overlooked to some extent. Even so, a limit of at most 4 drives per controller should be followed.

## Backup and Recovery Performance

In this section of the document, the performance aspects of various backup and recovery methods are analyzed, and some guidelines provided to help you choose the best method for your environment. We have focused our efforts on the software products discussed earlier: SQL Server 6.x and its native backup capabilities[40], and ARCserve 6.0 for Windows NT, which works in conjunction with SQL Server using the Backup Agent component. To represent backup throughput, the GB/hour metric has been chosen, and a MB/sec rate may occasionally be referenced (conversion is GB/hr = MB/sec * 3.6).

**DISCLAIMER:** The performance numbers presented in the following sections are to be used in relative comparisons between different configurations, and they are included to give the reader an idea which method and/or configuration achieves better throughput rates. The throughput rates can vary with the nature of data being backed up and with other variables. This is particularly true when using data compression. The nature of data being backed up determines the effectiveness of data compression algorithms, and in turn will affect the backup throughputs. Finally, these results were achieved using a dedicated SQL Server system experiencing no other application activity, and with no user activity occurring on the database(s) under test, unless otherwise stated[41].

### *Performance of Database Backup on Local Server*

This section offers a performance analysis of on-line database backups, or dumps, when done to storage devices located on the same (local) server. These types of backups usually result in the best performance, providing archival of data in the least amount of time. We will look at backup to various storage devices including disk arrays, 15/30-GB DLT tape, and 35/70-GB DLT tape. Also, different methods of implementing local database backup will be considered, such as using SQL Server striped dumps, dumping multiple databases concurrently, and using a third party application and backup agent.

#### System Configuration

The hardware we used for the performance testing conducted for this section consisted of the following:

- Compaq Proliant 5000 Server with 4 P6/166 processor boards.
- Compaq SMART-2/PCI Drive Array controllers.[42]
- Compaq 32-bit Wide-Ultra SCSI PCI controllers.[43]
- Compaq 15/30-GB and 35/70-GB DLT tape drives.
- Compaq DLT Tape Array and DLT Tape Array II cabinets.[44]

---

[40] Results including Microsoft SQL Server 4.21x were covered in the previous paper, *Compaq Backup and Recovery with Microsoft SQL Server* (4.21x), document #250A/0894.

[41] See section entitled **SQL Server Database Backup with User Activity**.

[42] For peak performance in Pentium Pro systems such as the Proliant 5000, it is important to ensure that a Wide-Ultra (SCSI) capable Smart-2/P Array controller is used, as opposed to earlier, Fast-Wide only versions. This is because of differences in the SCSI chipsets (53C875 vs. 53C825) that affect PCI bus utilization and thus data transfer rates. Note however, that we are using external storage cabinets for the disk drives, and so the controller actually operates in Fast-Wide mode as opposed to Ultra mode.

[43] For peak performance in Pentium Pro systems such as the Proliant 5000, it is important to ensure that a Wide-Ultra SCSI controller is used, as opposed to earlier, Fast-Wide versions. This pertains to the same issue encountered with the Smart-2/P controllers (see previous footnote), and was explained in the second chapter under the section entitled **Interface**.

[44] DLT Tape Array cabinets have Fast-SCSI-2 interfaces and hold 10/20-GB or 15/30-GB DLT drives. DLT Tape Array II cabinets have Wide-SCSI interfaces and hold 35/70-GB DLT drives.

- Compaq 4.3 GB Fast-Wide SCSI-2 disk drives (7200 RPM).
- Compaq External Storage cabinets for Fast-Wide disk drives.

The system was set up as follows:

- ❑ All system boot and binary files were installed on a single, internal hard drive on the integrated SCSI-2 controller.
- ❑ The test database was created on a dedicated logical volume (array) on the Smart-2 controller to isolate the I/O occurring to and from the data files, and spanned both SCSI channels on the controller. RAID level 5, which stripes the data across the disks in the volume along with parity information, was used for fault tolerance on this array. A smaller RAID-1 volume held the 'master' database and the transaction log file for the test database. Note that the transaction log dump was not measured as part of the database dump tests (see section on **Calculating Database Dump Throughput**).
- ❑ All 15/30-GB DLT Tape drives used for testing database dump to tape devices were attached either to the integrated Fast-Wide SCSI-2 controller or to additional Wide-Ultra SCSI-2 controllers[45], with no more than 2 drives per controller.
- ❑ All 35/70-GB DLT Tape drives used for the tests were attached to Wide-Ultra SCSI-2 controllers, 2 drives per controller.

**Potential Bottlenecks to Throughput**

Bottlenecks that can impede overall throughput in the system model include:

Source Side (Server):

- CPU - insufficient CPU cycles to process I/O requests or other backup related activity.
- System Bus(es) - saturation of peripheral bus-to-memory path when multiple high-throughput peripherals are used. Possible saturation of processor-to-memory path on SMP machines in CPU-intensive environment (not usually a concern in backup environments).
- Peripheral (expansion) Bus - saturation of EISA (16.6/33 MB/sec[46]) or PCI bus (66/132 MB/sec) interfacing to peripherals (disk controllers).
- Disk Controller Bus(es) - saturation of Fast-Wide SCSI-2 bus(es) (10/20 MB/sec[47]) on Smart-2 controller which contains the database volume.
- Disks - insufficient spindles in database volume to service required number of read requests from application.
- Software - serialization or contention points in application code, insufficient threads in application, etc.

Destination Side:

---

[45] As mentioned in the section referenced by the previous footnote, the integrated Fast-Wide SCSI-2 controller may be used in conjunction with additional, Wide-Ultra controllers without suffering an overall loss of throughput.

[46] Numbers in parenthesis indicate maximum regular and burst mode data transfer rates. Actual rate at saturation is usually lower.

[47] Numbers in parenthesis indicate maximum asynchronous and synchronous data transfer rates. Actual rate at saturation is usually lower.

- Device Controller Bus(es) - saturation of Fast-Wide SCSI-2 bus(es) (10/20 MB/sec) on 32-Bit SCSI-2 controller or Smart-2/P Array controller to which tape or disk dump devices interface.[48]
- Dump Device - limited capability of tape or disk device to service write requests (1.25/2.50 MB/sec[49] for DLT Tape drives).

Intermediate (Link):

- Network - insufficient bandwidth (1.25/12.5 MB/sec[50]) in the network cable.
- NIC Cards - limited capability of network controllers (at server or client) to process I/O requests to/from the network.

**A Word on Sequential I/O**

Backup processes generally involve sequential I/O meaning that successive read (or write) requests are issued for blocks of data that are stored contiguously on disk. Sequential requests can be processed by disk drives in rapid succession, because head seek times are minimal, and so each drive can be made to deliver very high throughput depending on the size of the requests (larger size requests yield higher throughput) and on the spin rate (RPM) of the drives. Sequential reads also allow for the use of 'read-ahead' algorithms in the hardware (like that in the Smart-2 controller) or in the application. Thus, when planning a strategy for optimal backup performance, it is important to remember that sequential I/O to the disk drives be maintained whenever possible. This concept will be stressed in this paper whenever applicable.

**Single Tape Device Dump**

Chart-3 represents results of SQL Server dumps to various single tape devices. In this section, the dump devices consist of a 15/30-GB DLT tape drive and a 35/70-GB DLT tape drive. A 10/20-GB DLT tape drive was not tested because its performance is identical to that of the 15/30-GB DLT. SQL Server versions 6.0 and 6.5 were both tested with each of the DLT drives, with Windows NT 4.0 providing the operating system platform.[51]

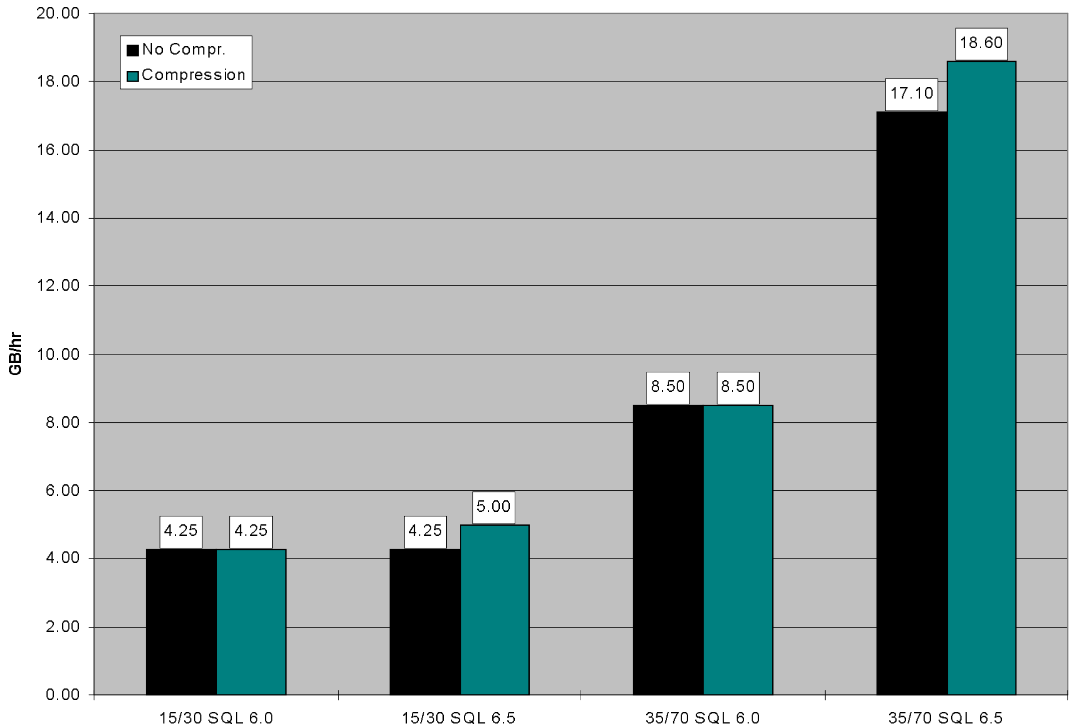
[48] We have recommended that Wide-Ultra SCSI controllers be used for reasons explained earlier. However, these controllers will operate in Fast-Wide mode with wide SCSI devices, or Fast-SCSI-2 mode with narrow devices.

[49] Numbers indicate data transfer rates with compression disabled and with 2:1 compression.

[50] Numbers in parenthesis indicate maximum data transfer rates for 10BaseT and 100BaseT ethernet. Actual rate is lower.

[51] We have not observed a significant difference in performance between Windows NT versions 4.0 and 3.51 for the purposes of these tape backup tests.

**Chart 3 - Dump to Single Tape Device**




Looking at the result sets, the steepest differences can be seen when comparing 35/70-GB DLT performance to 15/30-GB DLT performance: the 35/70 is almost 4x faster. This vast difference is understood when considering the hardware enhancements in the 35/70 vs. the 15/30, such as twice the number of simultaneous read/write channels (4 vs. 2), increased tape speed, increased tracks-per-inch on the media, enhanced interface (Fast-Wide SCSI-2), and a larger on-board buffer cache (8 MB vs. 2 MB).

Other differences in performance are seen when comparing SQL Server 6.0 results with SQL Server 6.5 results. The explanation for this essentially comes down to the issue of tape block size (please see the earlier section entitled **Block Size**). The larger read/write blocks specified by SQL Server 6.5 allow the DLT to record the data at a much faster rate **-** especially in the case of the higher-speed 35/70-GB drives where SQL 6.5 performs twice as fast. The smaller block size used by SQL Server 6.0 also prevents it from realizing any benefit from the hardware compression feature of the DLT drives.

With SQL Server 6.5 we can realize the benefits of hardware compression. From the above results however, these benefits seem limited since the performance with compression only slightly exceeds that without compression (the non-compressed throughput of the DLT is specified at 4.5 GB/hr and 18.0 GB/hr for the 15/30-GB and 35/70-GB drives, respectively). We must remember however, that the data used for our testing lends itself to compression less than would the typical numeric and textual data found in most production databases (our database was loaded with *randomly* generated numbers and text), therefore providing very conservative results. The compression achieved by the tape drive was measured to be around 15% (a 1.17:1 ratio); a greater increase in both capacity and throughput could be expected with a more compressible data set. Users should test the "compressability" of their own data using a representative data set[52].

---

[52] See the earlier section on **Estimating Hardware Compression Ratios.**

## Establishing Maximum Throughput at the Server - Null Device Dumps

Previously, for single-device dumps all reads from the disk array occurred sequentially by one thread. Now as we add dump devices SQL Server allocates additional threads from the backup threads pool[53], assigning one thread per dump device. The threads process data in round-robin order, with each thread reading one extent[54] at a time, skipping unallocated extents. The read pattern from the array as a whole remains essentially sequential in nature.

Also for single device-dumps, the overall throughput we could hope to achieve was limited by the speed of the storage device. When doing a dump operation to multiple, striped devices however, the throughput bottleneck may be shifted from the destination end (i.e: the storage devices), to the source end (the data drives), depending upon the number and speed of the backup devices used. Furthermore, the read capability of SQL Server is increased because there are now multiple threads reading from the database files, with each thread allocated its own in-memory buffer. We can attempt to compensate for this increased demand for asynchronous reads from the disk subsystem by adding a second controller - but another controller will increase read capability only if the two controllers are combined using a software array such as a Windows NT stripe set (please reference the earlier section entitled **Database Layout Considerations**).

For this section, the following additions were made to the server configuration:

- ❑ An additional Smart-2 controller was added to the primary PCI bus in our system.
- ❑ Each of the two SCSI channels (or 'SCSI-ports') on both of the Smart-2 controllers was supplied with four Fast-Wide SCSI-2 disk drives. Four drives were found to be sufficient to saturate a single SCSI channel when performing large size sequential I/O such as that performed in our testing.
- ❑ For tests in which the entire database resided on one Smart-2 controller, a single logical volume (array) using RAID-5 fault tolerance was created to hold the database, so that the data and parity is striped across the drives on both SCSI-ports using a 16KB striping factor. Using RAID-0 could have yielded results up to 5% slower for this configuration, because the 128KB striping factor used with RAID-0 corresponds less optimally to the 16KB sequential reads requested by SQL Server during database dumps. Using RAID-1 on the data volume could require twice the number of drives to achieve similar results.
- ❑ For tests in which the database was laid out across two Smart-2 controllers, 2 logical volumes using RAID-5 were created on each of the controllers (1 per SCSI-port). A stripe set using Windows NT Disk Administrator was then implemented to combine all 4 logical volumes.[55]

In this section we will attempt to determine the maximum read throughput achievable during a database dump on a Proliant 5000 system. In large, this involves increasing the read capacity of the disk subsystem by extending the database array across multiple SCSI channels and controllers. Lack of any throughput bottleneck at the receiving end was guaranteed by dumping to 'null' devices[56] (which act as 'bit buckets' for the data) so that the system is only reading (not writing) data. After having thus determined the throughput at the source end, we can then proceed to find the "optimal throughput" at the receiving end in the sections that follow.

---

[53] The number of threads reserved for dump and load operations is determined by the **backup_threads** parameter, and should be set equal to the maximum number of dump devices being used simultaneously.

[54] An extent is a SQL Server storage structure consisting of eight 2KB data pages.

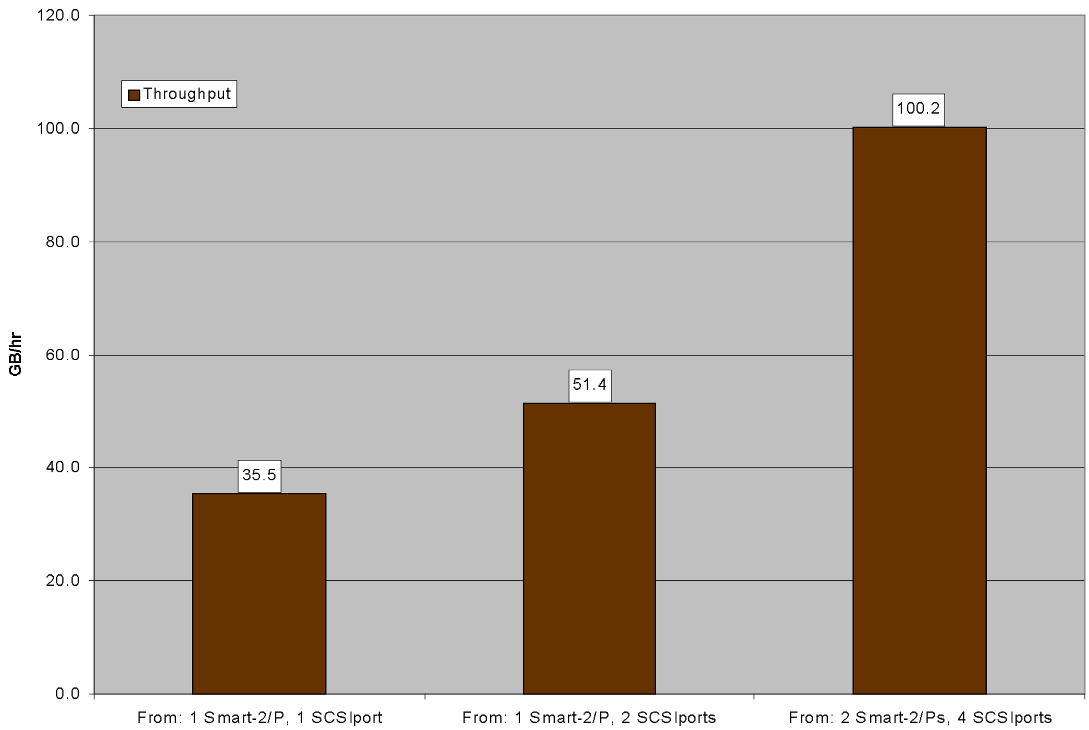
[55] For this configuration the striping factor for both Smart-2 controllers was changed to 128KB, which works better with the 64KB stripe size used by Windows NT striping. The default striping factor chosen by the Smart-2 for RAID-5 arrays is 16KB, but this setting resulted in up to 25% lower performance for our tests.

[56] The 'diskdump' device in SQL Server is a 'null' device.

The tests were done as striped dumps using two null devices (2 threads) for the first test, four devices (4 threads) for the second test, and six devices (6 threads) for third test. The results are shown below:

**Chart 4 - Null Device Dumps : Reading Data from Disk Arrays**




In each case the throughput is determined by the rate of read requests to the disk drives, and is limited by a combination of controller saturation (including SCSI bus bandwidth) and software overhead. Thus, **throughput increases as we increase the number of controllers and SCSI channels used**, until a maximum rate of 100 GB/hr is seen with the database implemented across both channels of two Smart-2 controllers. Note that both controllers were configured on a single PCI bus; the server has a second PCI bus which is still idle. In many system layouts however, multiple disk controllers are not combined into a single array using software striping, so that SQL Server will rarely be reading data from more than one controller at a time during a dump. Therefore the maximum dump throughput achieved will usually be limited to around 50 GB/hr (second bar in above chart). We will expect that to be the case in the following sections, where striped dumps are done to various hardware storage devices.

**Striped Device Dumps - Smart-2 Disk Arrays**

SQL Server allows the use of "disk dump devices" for backup purposes, as well as tape. The key benefit of using disk storage is performance: a backup process (dump) to disk can often be completed very fast, especially if sending the data to a disk array controller that can spread the writes across multiple disk drives (spindles). Of course disk storage will always be limited by capacity, for the disk media cannot be conveniently replaced and stored like a tape cartridge. The disk solution should generally be used in conjunction with an off-line backup utility which can later transfer the dump image files onto tape, thus allowing normal database transactions to continue on the SQL Server with minimal time spent on dump activity. If the preservation period of your archives is relatively short however (i.e: the backup sets are completely replaced quite frequently), then a large, dedicated disk array could conceivably be used for storage of the data without an accompanying tape device[57]. In this case a fault tolerant array would be highly recommended, preferably on a separate controller from the one containing the actual database volume.

In this section backup data is sent to SQL Server disk dump devices, which are created on additional Smart-2/P Array controllers. For these types of dumps to local disk array controllers, the following modifications were made to the system configuration:

- External storage cabinets housing 5 Wide SCSI drives each were attached to each of the Smart-2/P controllers used to store the backup data. Smart-2 controllers used to hold the disk dump devices were placed on the secondary PCI bus of the Proliant 5000, while the controllers holding the database itself were placed on the primary PCI bus. In this way, system throughput was optimized because the load was split evenly across the peer PCI channels[58] (all data read into system memory from peripherals on PCI bus 1, all data written out from memory to peripherals on PCI bus 2).[59]
- A single disk array was configured on each SCSI port of each controller. A single disk dump device was created under SQL Server for each array, so that only one stream of writes will occur to any one array. This way, sequential writes are enforced on each array even though multiple dump devices are used. You can create more than one dump device file per array, but then the writes to this array will become randomized and performance may suffer (due to disk head contention).
- Backup ‘stripe sets’ were created with the dump devices. The SQL Server dump process uses one ‘thread’ per dump device, so that either 2 or 4 threads are being used depending on the number of devices. (In Chart-5, this works out to #threads = #SCSIports).
- Destination (dump) disk arrays were built using RAID-5 fault tolerance, then again with RAID-0 for better performance.
- A minimum 2:1 ratio is maintained at all times between the number of SCSIports/disks used for writes vs. for reads (i.e: there are always at least twice the number used for the dump devices as for the data array). This is to prevent any ‘funnel’ effect on the writes, since the Smart-2 can perform sequential reads faster than it can complete large, sequential writes.
- The on-board cache of the Smart-2 controller(s) used for the dump devices was dedicated for write operations[60].

Using the above guidelines, performance was measured for three different system layouts - using both the RAID-0 and RAID-5 array dump devices (6 tests total). The results are shown in the below chart:

---

[57] The feasibility of using disk volumes for incremental backups is enhanced by SQL 6.x’s ability to append data to disk dump devices.

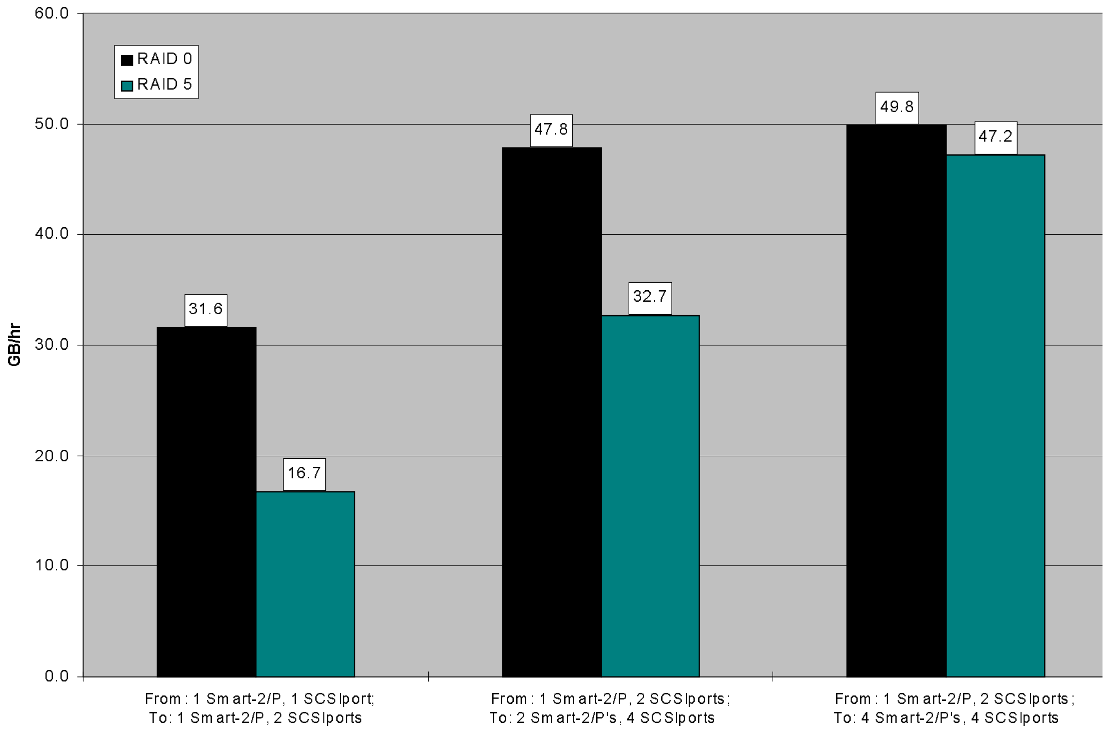
[58] Exception: for the last set of tests in Chart-10, 1 read controller and 2 write controllers were configured on each of the PCI buses (3 controllers per bus).

[59] If the peripheral controllers are placed so that all data transfers occur on a single PCI bus, throughput may be limited by the PCI bus bandwidth.

[60] The on-board controller cache was manually configured for 100% write caching (vs. read-ahead) using the Array Configuration Utility. However, this adjustment would be made dynamically by the controller logic if no read activity was detected on the volume.

**Chart 5 - Striped Dumps to Disk Arrays**




As expected, performance increases as the number of storage devices (controllers, SCSIports, spindles) used is increased. The performance trend when dumping to RAID-0 arrays however, differs from that seen with RAID-5 arrays. For the RAID-0 arrays, when the number of storage devices is doubled from the first test to the second - throughput jumps about 50% (from about 32 to 48 GB/hr). After that only a marginal increase is seen, because at this point the **limit of the source (read) controller to deliver data** is already reached. With a throughput of nearly 50 GB/hr on the third test however, we have reached the maximum throughput predicted in the previous section (allowing for a small additional overhead for write processing)[61].

Performance for the RAID-5 arrays at first lags significantly behind that of the RAID-0, but scales quickly as storage devices are added. The performance doubles as the number of storage devices are doubled from the first test to the second (from 16 to 32 GB/hr). This is because the throughput limitation of the source controller has not yet been reached - at this point our bottleneck is at the Smart-2 controllers used to write the data to; RAID-5 is very write-intensive and tends to saturate the data channels on the controller when large (64KB) writes requests are rapidly being issued[62]. From the second to the third tests, the number of controllers has been doubled again, so that the performance

[61] The throughput for the 3rd RAID-0 test may have yielded a higher number if SQL Server had been reading the database from two Smart-2 controllers combined into a Windows NT stripe set, as with the 3rd null device test in the previous section.

[62] Writes to a RAID level 5 disk array may involve up to 4x additional I/O. When transferring data to a RAID-5 volume, 4 physical I/O's (2 reads and 2 writes) must normally occur for every logical write request so that the parity information may be re-created. If purely sequential writes occur to the array however, the Smart-2 controller can sometimes recognize the contiguous data pattern and thus perform fewer additional I/O's for the parity (i.e: parity can be generated for each "stripe" across all drives instead of for each write to any one drive). For a discussion on the performance implications of RAID technology, see the *Configuring Compaq RAID Technology for Database Servers* white paper.

goes up almost 50% to 47 GB/hr which is close to what it was with RAID-0. The average CPU usage seen during the third set of tests (for both types of arrays) was around 60%.

**Striped Device Dumps - 15/30 DLT Tape Arrays**

In this section data is sent to multiple 15/30-GB DLT[63] tape drives, using the tape drive striping functionality of the SQL Server **dump database** process. For these types of dumps to multiple tape drives, the following considerations were taken into account:

- As before, the database was contained on a single RAID-5 disk array, using a single Smart-2/P controller (both SCSI-ports) which resided on the primary PCI bus of the Proliant 5000.
- All of the DLT drives resided in Compaq DLT Tape Array cabinets (4 drives per cabinet). Each cabinet was interfaced to 2 controllers (the embedded Fast-Wide SCSI-2/P controller or additional Wide-Ultra SCSI-2/P cards). Up to 12 tape drives and 6 controllers were used in the testing, with no more than 2 drives attached to any one controller.[64]
- All tape drives are used as part of a single tape device 'stripe set'. Since up to **12** tape drives were used, the **backup_threads** parameter was set accordingly, as the SQL Server dump process uses one thread per device.
- Compression was enabled on the tape drives, providing an approximately 15% increase in throughput vs. non-compressed. A database containing data that lends itself differently to compression could yield throughput somewhat higher or lower than that displayed below.

Using the above guidelines, performance was measured using backup sets of 1 to 12 DLT drives. Since we expect to get a high throughput rate from 12 DLT's, we monitor the hardware channels on the system (PCI bus[65] and CPU usage) to ensure that a performance bottleneck is not encountered elsewhere. The results are shown in the following chart:

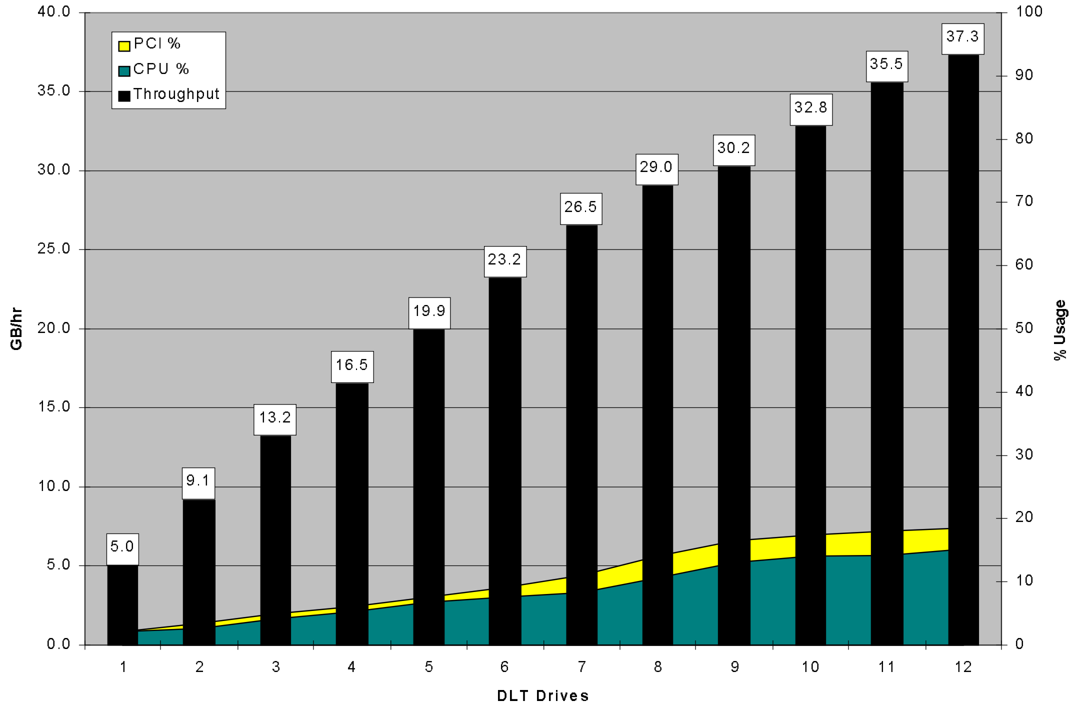
[63] The results in this section can be applied to 10/20-GB DLT tape drives as well, since the performance of these drives is the same as that of the 15/30 drives.

[64] The 8-bit, asynchronous mode SCSI transfers done by the DLT will allow us to practically use no more than about 3 MB/sec of the SCSI bus bandwidth, and so throughput limitations are encountered beyond 2 drives on a controller. With 12 drives however, the throughput maintained to each individual drive falls below even 1 MB/sec, so that placing 3 drives per controller can be considered.

[65] PCI bus usage was monitored using special Performance Monitor counters developed by Compaq, which are available through the Compaq Resource Kit for Windows NT. The PCI % utilization shown in the chart was taken as an average of the load observed between the 2 peer PCI buses.

**Chart 6 - Striped Dumps to 15/30 DLT Tape Drives**




Starting with 1 drive yields the 5.00 GB/hr throughput seen before with a single drive. As Chart-6 shows, the expected increase in throughput is then seen as additional drives are added to the stripe set. Looking at the chart, we notice a general trend of "diminishing returns" whereby the **gain in throughput achieved by adding drives decreases successively** as more drives are added. Earlier, we predicted that the ability of the Smart-2 controller to deliver data would restrict the system to under 50 GB/hr. However, this trend of decreasing gains actually begins much earlier, and continues so that the system never reaches the 50 GB/hr maximum. From 1 to 8 drives our scalablility (performance increase) is actually quite good - an average of 3.5 GB/hr increase for each additional drive. From 9 to 12 drives the scalability is more modest - averaging to a 2.0 GB/hr increase per additional drive. We therefore recommend that **no more than 8** 15/30-GB (or 10/20-GB) DLT drives be used in a single system for this type of backup activity. Production environments that wish to use the SQL Server dump but that require a backup rate 30GB/hr or greater to meet their time constraints, should consider a solution using either disk drives (previous section) or 35/70-GB DLT drives (next section).

The reason for the non-linear scalability probably pertains to a combination of hardware and application behavior. Ideal scalability in complex systems is rarely achieved, and in this case data traveling from one media (disk) to another media (tape) has to go through numerous intermediate hardware paths (SCSI bus, PCI bus, system memory, etc.). Although we can see from the above chart that neither the system processors nor the PCI bus is close to saturation, there are still latencies associated between each of the hardware paths which can combine to cause overall system scalability to fall-off as load increases. Furthermore, we should remember that although multiple threads of operation are being used by SQL Server to deliver data to multiple tape drives, all of the threads are reading from a *single* source (one database) and are forced to alternate between each other so that a thread which has requested data for its tape drive cannot again submit a read request until all other threads have done so. This 'round-robin' processing among the different device threads brings about a certain amount of serialization in the application that affects scalability.

Note that when doing a SQL Server striped dump it is possible to combine different tape drives, or even different types of dump devices, into a stripe set. We do not recommend however, that DLT drives be combined with other, slower types of tape drives because this will cause the performance of the DLT drives to be reduced to that of the slowest drive in the set.

Finally, as mentioned in previous sections, the compressability of the data in your database can affect the performance numbers seen when dumping to tape drives supporting hardware compression. We have found however, that performance differences due to compression become less apparent as more DLT drives are striped together to form a single high-speed tape array.

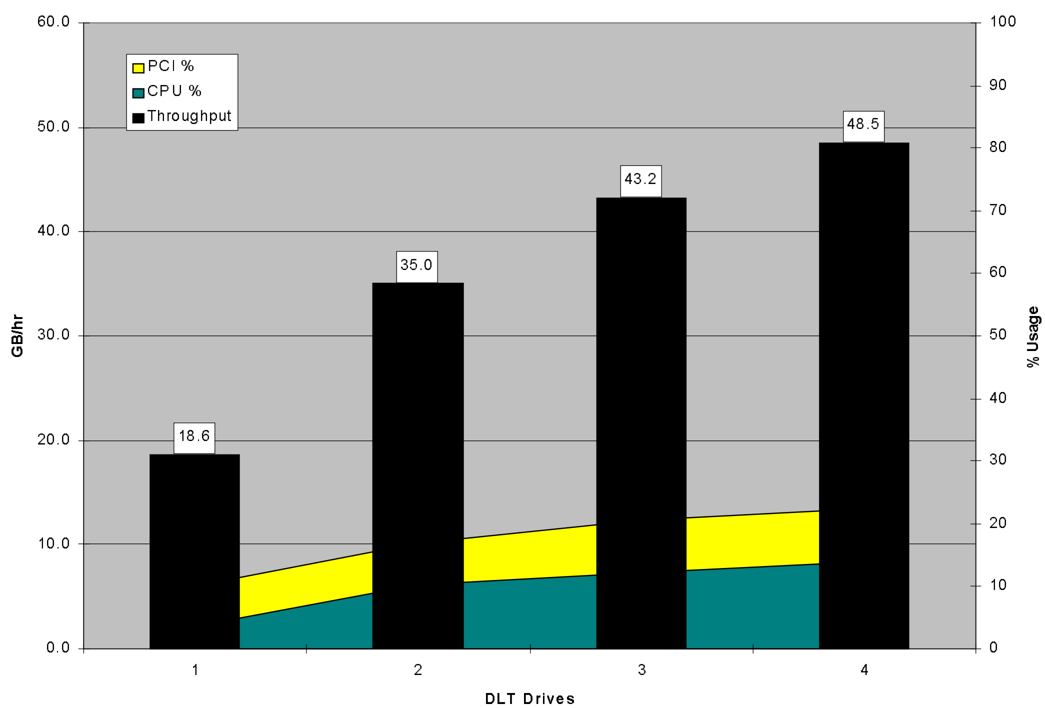
## Striped Device Dumps - 35/70 DLT Tape Arrays

In this section data is sent to multiple 35/70-GB DLT tape drives, using the tape drive striping functionality of the SQL Server **dump database** process. The 35/70 DLT offers not only greater capacity but also greater performance than the 10/20 and 15/30 DLT's; at the start of this chapter we demonstrated this performance increase to be almost 4x. For these tests, the following changes were made to the system configuration:

- All of the 15/30-GB DLT drives were removed from the system. Four 35/70-GB DLT drives were attached to two of the Wide-Ultra SCSI-2/P controllers (two drives 'daisy chained' to each controller) on the secondary PCI bus.
- All tape drives are used as part of a single tape device 'stripe set'. Since up to **4** tape drives were used, the **backup_threads** parameter was set accordingly, as the SQL Server dump process uses one thread per device.
- Compression was enabled on the tape drives. A database containing data that lends itself differently to compression could yield throughput somewhat higher or lower than that displayed below.

Performance was measured using backup sets of 1 to 4 DLT drives. The throughput results are shown in the following chart, along with certain hardware usage counters:

**Chart 7 - Striped Dumps to 35/70 DLT Tape Drives**




The increased performance of the 35/70-GB DLT drives results in far fewer drives being required to achieve the same performance seen with the 15/30-GB DLT's; with only 2 drives the dump throughput is about the same as with 11 of the 15/30's. The throughput has also doubled from the first drive to the second, showing SQL Server's ability to take advantage of a second high-speed drive. After the second drive, further performance increase is less pronounced, as **the bottleneck begins to shift from the tape drives to the disk subsystem**. With four drives a throughput of 48.5 GB/hr is achieved, which is close to the maximum read rate observed earlier for a single Smart-2 Array controller in this environment.

The situation encountered here, with respect to the highest backup rate observed, is similar to what was seen when sending data to disk storage. Indeed, a single 35/70-GB DLT drive can receive data at a rate about the same as a RAID-5 disk array on a single Smart-2 SCSIport (see Chart-5). Without going to a configuration involving multiple controllers interleaved with software RAID[66], we can consider these results as approaching the *maximum throughput achievable with a single SQL Server database dump* process on the current hardware.[67] An even higher overall throughput can be realized from the system when backing up more than one database in parallel, which will be demonstrated in the next section.

A note about so-called "maximum" system throughput: In the previous section we discussed hardware latencies that can contribute to a decrease in system performance scalability. We should also point out that I/O requests from the application to the hardware have to go through numerous software layers as well (operating system, tape driver, miniport driver, etc), any one or a combination of which could introduce software latencies or contention points when multiple such requests are being issued for multiple devices. The software environment is significant in determining a system's performance, and the same hardware could produce different results when driven by different software.

### Concurrent Database Dumps

This section discusses dumping multiple databases concurrently; that is, having SQL Server dump more than one database at the same time (in parallel). If more than one database resides on the same SQL Server, this strategy can be used to increase the backup capability (overall throughput) of the server. When employing multiple devices to perform such concurrent database dumps, be aware of the following considerations:

- You can perform one dump (transaction log or database) per dump device. To dump two databases simultaneously to tape, you need to have installed two tape drives. These drives can coexist on the same controller and in the case of DLT drives, the recommended limit of two such drives per controller (discussed earlier) relates here as well. The same applies to load operations.
- Previous versions (4.21x) of SQL Server upon startup allocated only three internal buffers dedicated to dump and load operations. If all three buffers were consumed, as with three simultaneous dump or load processes, attempting a fourth dump or load would yield little gain in overall throughput. SQL Server 6.x now provides the capability for up to 32 simultaneous dump or load processes, each with its own backup buffer. The **backup_threads** parameter must be set to equal the maximum number of devices that will be used at one time, as in the case of striped dumps.
- A single database cannot be dumped concurrently to more than one device at a time. Only after the first dump command completes can another dump for the same database begin.
- In order to effectively employ multiple concurrent dumps, sequential I/O must be preserved on both the source and destination volumes so as not to introduce disk head contention.

---

[66] See earlier section entitled **Establishing Maximum Throughput at the Server**.

[67] When SQL Server 6.5 database dump tests were run on previous generation (EISA-based) Compaq servers (i.e: the Proliant 4500), the maximum throughput achieved was about 25 GB/hr, due mainly to limitations imposed by EISA bus bandwidth. This throughput was about half of what is seen on the PCI-based Proliant 5000. Future hardware from Compaq will feature technology enhancements that may push the current numbers even higher.

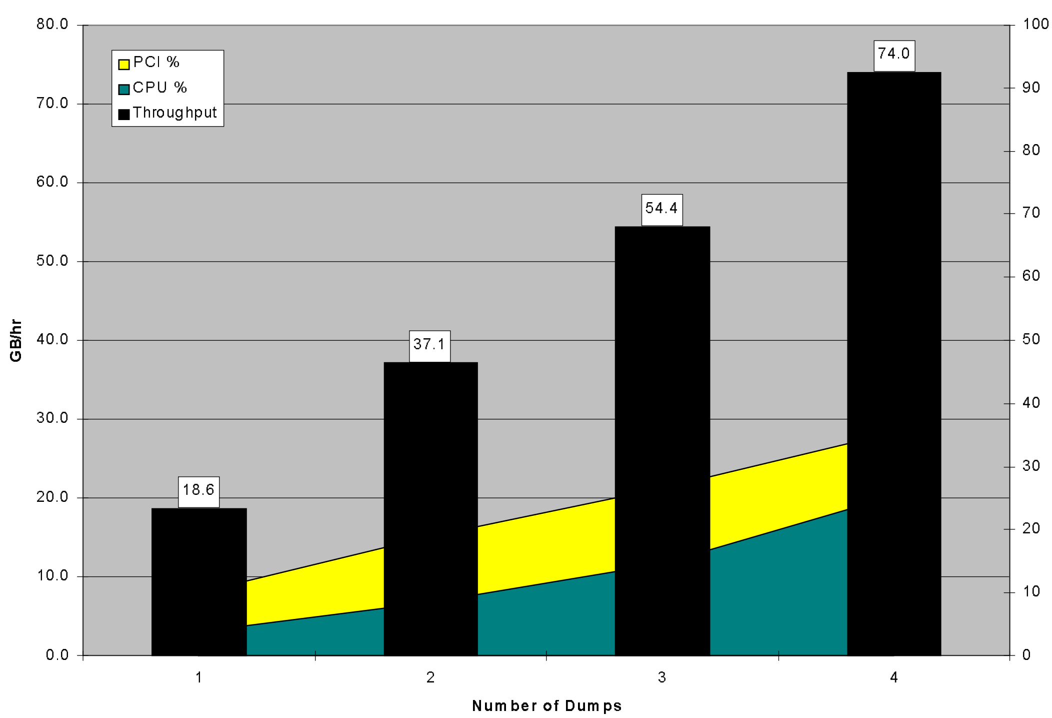
The final point above deserves further discussion. When a single database is being dumped, the dump process begins reading the data at the start of the database file and continues reading successive pages that are stored more or less contiguously on disk (even in an array). When a second database residing on the same disk or disk array begins, the disk heads will have to go back and forth between the two locations, increasing seek time and slowing performance. Thus, the only way to retain steady, sequential read paths is to *isolate each database on its own volume*. The only exception to this principal, is for dumping two databases residing on the same mirrored (RAID-1) volume. In this case, the Smart-2 SCSI Array controller can effectively keep the reads sequential on each set of mirrored drives by using one set to satisfy read requirements of the first dump and the other set for the second dump (known as "split seeks").

For the tests involved in this section, the system was configured as follows:

- ❑ A second Smart-2/P Array controller was installed in the system. A four spindle RAID-5 array was created on each SCSI-port of both controllers, providing four logical drive volumes. Four identical databases were then created, one on each drive volume.
- ❑ The four 35/70-GB DLT Tape drives configured across two Wide-Ultra SCSI controllers (2 drives per controller) were retained from the previous tests. Thus there will be one tape drive to receive data from each of the four databases.
- ❑ The controllers that will be sending data (the Smart-2 SCSI) are on the system's primary PCI bus, and the controllers that will be receiving data (the Wide-Ultra SCSI) are on the second PCI bus. This configuration is important to ensure load balancing so that PCI bus saturation will not be encountered.

Using the above configuration, we performed dumps of from one to four databases simultaneously, each to its own 35/70 DLT drive, while monitoring the usual hardware usage percentages. The results are shown below:

**Chart 8 - Concurrent Dumps of Multiple Databases to 35/70 DLT Tape Drives**




The benefits of the concurrent dump strategy are quickly realized as additional databases are dumped. When dumping one database to a single drive the throughput is 18.6 GB/hr, and when dumping two databases to two different drives the combined throughput is 74 GB/hr - about a 4x increase. Another way to look at this is to consider that **dumping four databases concurrently will take only about one-fourth as long as dumping them sequentially (back-to-back)**. Furthermore, by doing enough concurrent operations to DLT tape, the overall throughput reaches a rate (74 GB/hr) higher than what we've seen before. The hardware utilization at this point is around 26% CPU and 36% PCI, so that bandwidth yet remains in the system for additional jobs.

Since concurrent dumps involve multiple databases, each of which can reside on the same or different disk controllers, we are no longer bound by the read rate of a single controller and so achieve a very high overall throughput. Furthermore, since each dump operation is separate and has its own dedicated I/O devices (disk array and tape drive), the operations are truly parallel and display near-linear scalability (increase in performance).

Finally, note that concurrent dumps can also be done as striped dumps, so that SQL Server is dumping multiple databases with each database dump writing to multiple tape drives. Several different combinations of drives-to-databases could be possible for such "concurrent, striped dumps". For example, in the above system we replaced the four 35/70-GB DLT drives with twelve 15/30-GB DLT drives and configured backup 'stripe sets' of three tape drives for each database. When all four databases were dumped concurrently, an overall throughput exceeding 50 GB/hr was observed.

### Database Backup to 15/30-GB DLT Tape RAID - ARCserve for Windows NT

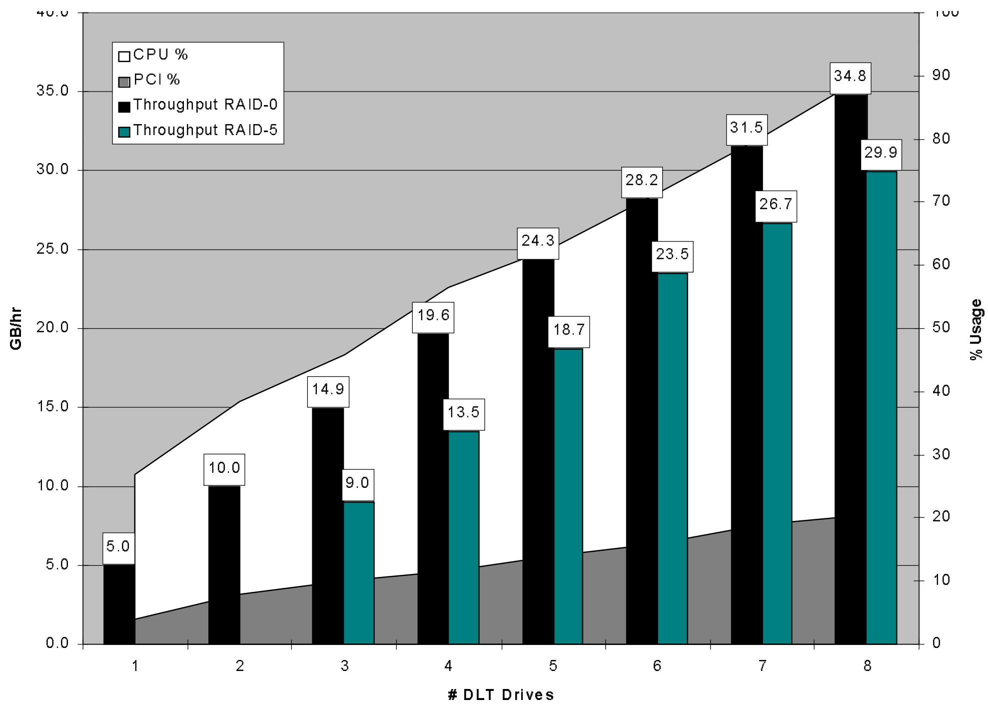
The analysis done in this section will involve the use of 'third party' products to supplement the SQL Server database backup functionality: ARCserve 6.0 for Windows NT, the ARCserve Backup Agent for SQL Server option, and the ARCserve RAID option. Details about the features and operational characteristics of these products were covered in an earlier section: **Online Backup Considerations with ARCserve for Windows NT**.

All backup tests discussed in this section were performed by sending data from an online SQL Server database to 15/30-GB DLT tape drives on same (local) server. As with previous tests, no user transaction load is occurring on the database being backed up, or on any other database on the system. The ARCserve applications act as a 'go-between' from SQL Server to the tape subsystem. For these types of operations utilizing ARCserve to backup databases to multiple tape drives, the following considerations were taken into account:

- The database was contained on a RAID-5 disk array, using a single Smart-2/P controller (both SCSI ports) which resided on the primary PCI bus of the Proliant 5000.
- All of the DLT drives resided in Compaq DLT Tape Array cabinets (4 drives per cabinet). Each cabinet was cabled to 2 controllers (the embedded Fast-Wide SCSI-2/P controller or additional Wide-Ultra SCSI-2/P cards), so that each controller interfaced to 2 drives. Up to 8 tape drives and 4 controllers were used in the testing.
- All tape drives are grouped into a single 'array' using the ARCserve RAID Option. Tests were conducted using from 1 to 8 tape drives in a RAID-0 array, and then with 3 to 8 tape drives in a RAID-5 (fault tolerant) array.
- Compression was enabled on the tape drives, providing as much as a 15% increase in throughput vs. non-compressed. A database containing data that lends itself differently to compression could yield throughput higher or lower than that displayed below, although such differences become less apparent as more DLT drives are striped together to form a single high-speed tape array.

Using the above guidelines, performance was measured while monitoring processor usage and PCI bus usage (avg. of 2 PCI). The throughput results for all tests, along with the hardware usage percentages for the RAID-0 tests, are displayed in the following chart:

**Chart 9 - Database Backup to 15/30 DLT Tape RAID Using ARCserve**




As we can see, using the ARCserve software in conjunction with SQL Server yields excellent throughput and scalability, both with RAID-0 and RAID-5 tape arrays. The average increase in throughput when adding a tape drive to either type of array is around 4.25 GB/hr. As with disk arrays, RAID-5 tape performance falls below RAID-0 performance; the average difference between the two is about 5.3 GB/hr. In fact, the throughput for an array of 'n' tape drives in a RAID-5 array is close to the performance of 'n-1' drives in a RAID-0 array. This makes sense when we consider that for every 'n' blocks of data (1 per drive) sent to the RAID-0 array in a single "stripe", 'n-1' blocks of data and 1 block of parity are sent to the RAID-5 array. There is some additional performance overhead in RAID-5, probably because the program has to calculate the parity block (a boolean XOR operation on the data blocks) and interleave it among the drives. Although not shown in the chart, the RAID-5 operations had a CPU% usage that was slightly more than for the RAID-0 operations, and a PCI% usage that was slightly less. The fault tolerance provided for the data set with the RAID-5 option however, may be worth the performance penalty if archiving data in a mission-critical environment.

The next comparison that should be made here is with the results achieved from the native SQL Server striped dumps (see Chart-6), which can be likened to the ARCserve RAID-0 array backups. Backup operations to **ARCserve RAID-0 tape arrays outperform dump operations to similarly sized SQL Server tape stripe sets**, especially as more tape drives are added. With 8 DLT drives, ARCserve can backup a SQL Server database almost as fast as SQL Server can with 11 DLT drives (about 35 GB/hr). The tape block size used by both applications is the same (64KB).

This performance variance is interesting to note, especially since the ARCserve database backup agent itself makes use of a SQL Server striped dump operation. The difference in the two procedures however, is that with a SQL Server native striped dump, each thread reads data pages from disk into a memory buffer, and from there writes the data out to each of the tape drives. With an ARCserve RAID dump, each of the dump threads writes the data to a local named pipe created by the backup agent. The backup agent provides a thread for each pipe to retrieve the data and send it to the ARCserve job process, where it is written to the tape drives. Apparently the use of the named pipes mechanism allows the ARCserve backup agent to process the incoming data and send it out to tape faster than the native SQL Server process can. However, this mechanism also demands much more CPU power; with eight DLT drives our four Pentium Pro/166 processors were almost saturated (CPU usage split between the SQL Server and ARCserve threads). This high load on the system CPU's could cause performance degradation to other application activity on the system, or to user activity in other databases on the system, should the backup job be run during production hours[68].

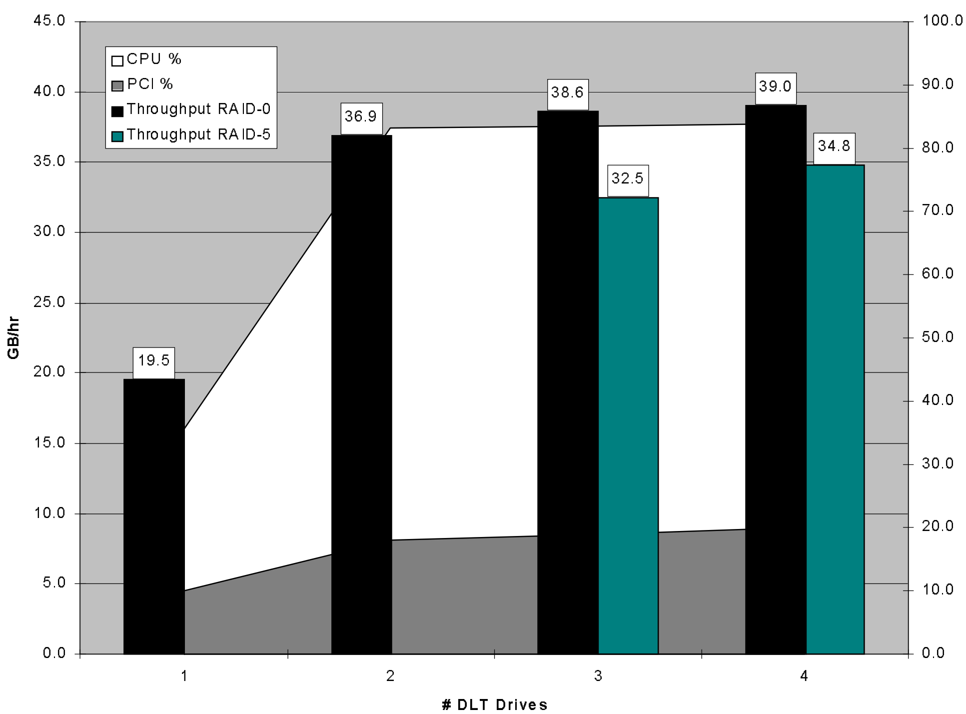
**Database Backup to 35/70-GB DLT Tape RAID - ARCserve for Windows NT**

We now continue the performance evaluation of the ARCserve products for SQL Server online database backup, but with the higher-speed 35/70-GB DLT tape drives. For these tests, the following changes were made to the system configuration:

- All of the 15/30-GB DLT drives were removed from the system. Four 35/70-GB DLT drives were attached to two of the Wide-Ultra SCSI-2/P controllers (two drives 'daisy chained' to each controller) on the secondary PCI bus.
- All tape drives are grouped into a single 'array' using the ARCserve RAID Option. Tests were conducted using from 1 to 4 tape drives in a RAID-0 array, and then with 3 to 4 tape drives in a RAID-5 (fault tolerant) array.
- Compression was enabled on the tape drives. A database containing data that lends itself differently to compression could yield throughput somewhat higher or lower than that displayed below.

Performance was measured using backup sets of 1 to 4 DLT drives. The throughput results are shown in the following chart, along with hardware usage counters for the RAID-0 tests:

**Chart 10 - Database Backup to 35/70 DLT Tape RAID Using ARCserve**




In the previous section, it was observed that using the ARCserve products (including the SQL Server backup agent and the RAID option) could offer better performance than using only the native SQL

[68] See section on **SQL Server Database Backup with User Activity**.

Server backup functionality, with 15/30-GB DLT's. From the above chart, we see that this is not the case with the 35/70-GB drives (compare to Chart-7). Using ARCserve yields good backup performance and scaling up to the second drive; after that almost no further gains are realized with additional drives.

The reasons for this performance "flat-line" come down to the high CPU utilization demanded by the database backup agent and SQL Server when performing this type of a database dump. **Saturation of the system processors restricts the backup throughput** to around 38 GB/hr with RAID-0 and 34 GB/hr with RAID-5, and interferes with the system's ability to process other activity (such as user transactions on another database) for the duration of the backup job. Given these results, it would follow that no more than two 35/70 DLT drives should be used in a database backup to an ARCserve RAID-0 tape array, and no more than three such drives in a RAID-5 tape array. Note however, that a system with more powerful processors than our four Pentium-Pro/166 modules, could conceivably yield better performance with more drives.

Finally, it should be noted that we did not attempt to test a scenario involving concurrent backups of multiple databases to storage on the local server using ARCserve for Windows NT, due to the high CPU use. We do however, recommend ARCserve (and its autochanger option) for implementing concurrent, unattended backup jobs to both drives inside the Compaq DLT 15-Cartridge Library[69]. Also, results of using the ARCserve products to perform concurrent backups of databases to a *remote* server, will be shown in the following sections.

[69] CPU usage for such a scenario would not be expected to exceed around 50% in a 4P/166 Proliant 5000.

## *Performance of Database Backup to Remote Server*

This section covers performance analysis of a different kind of backup strategy from the scenarios previously discussed. Remote backups involve backing up a database which resides on one server, by transferring data across a network interconnect to storage devices which reside on another server. This method allows for the 'pooling' of tape or disk storage devices on a central storage server, to which data is sent from other servers on the network. The need to have local backup devices on each server in the environment is thus eliminated. When multiple tape or disk drives exist on the storage server, the remote backup strategy can also be employed to backup multiple database servers simultaneously.

Remote backups can be initiated from each of the respective database servers using SQL Enterprise Manager, which can be likened to a data 'push' scenario. In this case the backup must be done to disk media on the destination (storage) server, for SQL Server 6.x cannot 'dump' a database to remote tape drives by itself. No application software is needed on the storage server, only Windows NT file shares accessible from each of the source servers. The remote backup device(s) must be added at the SQL Server(s) as a disk device using the full UNC path-name in form: **\\server\share\filename** (ex: \\backup1\dumps\DB.DAT). The MSSQLServer service should be configured to log on using a user account which has access to the remote share (via the Services applet in the Control Panel)[70]. Remote backups can also be initiated centrally from the storage server using the ARCserve Manager, which can be likened to a data 'pull' scenario. In this case the backup is done to tape media on the storage server, for ARCserve 6.0 cannot backup to disk devices. Each of the source servers must have the ARCserve Backup Agent for MS SQL Server installed so that their online databases are available for backup.[71] For more information on ARCserve and its add-on software, see the earlier section entitled **Online Backup Considerations with ARCserve for Windows NT.**

Database backup operations are highly throughput-intensive operations requiring fast data transfer rates. With the powerful systems and applications available to us, very fast backups can be achieved to high-speed storage devices connected locally, as we have already shown. In most remote backup scenarios however, the network devices end up becoming the bottleneck. Our goal should therefore be to alleviate the network bottleneck by using strategies involving high speed 100 Mb/s Ethernet (in addition to slower, 10 Mb/s Ethernet), multiple segments for concurrent backups, and multiple Network Interface Cards (NICs) per segment.

### System and Network Configuration

The hardware we used for the performance testing conducted for this section consisted of the following:

**Storage Server:**

- Compaq Proliant 5000 Server with 4 P6/166 processor boards.
- 4 Compaq 32-bit Fast-Wide SCSI-2 PCI controllers (C875).
- 1 Compaq SMART-2 Array PCI controller.
- 8 Compaq 15/30-GB DLT tape drives.
- 1 Compaq 35/70-GB DLT tape drive.

---

[70] Alternatively, the name of the remote share can be added to the "NullSessionShares" registry key under the path \\HKEY_LOCAL_MACHINE->CurrentControlSet->Services->LanmanServer->Parameters. For more information on setting up a network dump device, see the *SQL Server Administrator's Companion.*

[71] The actual database device files can be backed up to remote tape by ARCserve without the use of the Database Agent, but would require the SQL Server to be shut down. Such an off-line backup strategy is not tested in this paper.

- 2 Compaq DLT Tape Array cabinets.
- 1 Compaq External Storage cabinet with 5 Fast-Wide disk drives.
- 3 Compaq NetFlex-3 PCI 10/100 TX UTP cards.

This is the primary system under test. It was set up as follows:

- The system boot and binary files were installed on a single internal disk drive attached to the integrated Fast-Wide SCSI-2 controller. The external storage cabinet was attached to one of the Smart-2 ports, and configured as a RAID-5 volume for the disk dump tests. The external cabinet was later removed, and the Smart-2/P was removed to make room for Fast-Wide SCSI-2 controllers needed for the tape backup tests. No database was built on this system.
- Up to 4 NetFlex-3 NICs were added to the system, as multiple NICs were used for tests in which multiple remote databases were backed up in parallel. These cards are capable of detecting either 10 Mb or 100 Mb transfers, so could be used for both types of tests. All of the NetFlex-3's were placed on the system's primary PCI bus
- All DLT Tape drives used for the tape dump device tests were attached either to the integrated Fast-Wide SCSI-2/P controller or to additional Wide-Ultra SCSI controllers. SCSI controllers used for the tape drives were placed on the secondary PCI bus of the Proliant 5000, apart from the network cards which were placed on the primary PCI bus. In this way, system throughput was optimized because the load was split evenly across the peer PCI channels (all data transferred into the system from peripherals on PCI bus 1, all data written out from memory to peripherals on PCI bus 2).

**Database Server:**

- Compaq Proliant 5000 Server with 4 P6/166 processor boards.
- 1 Compaq SMART-2 Array PCI controller (C875).
- 1 Compaq External Storage cabinet with 5 Fast-Wide disk drives.
- 1 Compaq NetFlex-3 PCI 10/100 TX UTP card.

This was the same system used for previous tests, with some modifications. It was set up as follows:

- The system boot and binary files were installed on a Smart-2 volume on the internal drives, along with the transaction log file for the database. The transaction log was first truncated so as not to figure into the database backup throughput (transaction log dumps are analyzed later).
- The system boot and binary files were installed on a Smart-2/P volume composed of drives in the internal bays. The external storage cabinet was attached to the external Smart-2 port, configured as a RAID-5 volume to hold the database.
- All SCSI-2/P controllers and DLT tape drives were removed from the system and added to the storage server (see above).

For the later tests involving concurrent backups, another Proliant 5000 similar to the above was brought in to serve as a second database server. Two other machines were also configured as database servers, with the following hardware:

- Compaq Proliant 2500 Server with 1 P6/200 processor board.
- 1 Compaq SMART-2 Array PCI controller (C875).
- 1 Compaq External Storage cabinet with 4 Fast-Wide disk drives.
- 1 Compaq Integrated Netelligent 10/100 TX PCI card.

**Network:**

The network operating system used on all servers in this section was Windows NT 4.0. The network protocol selected was TCP/IP. Networking hardware included:

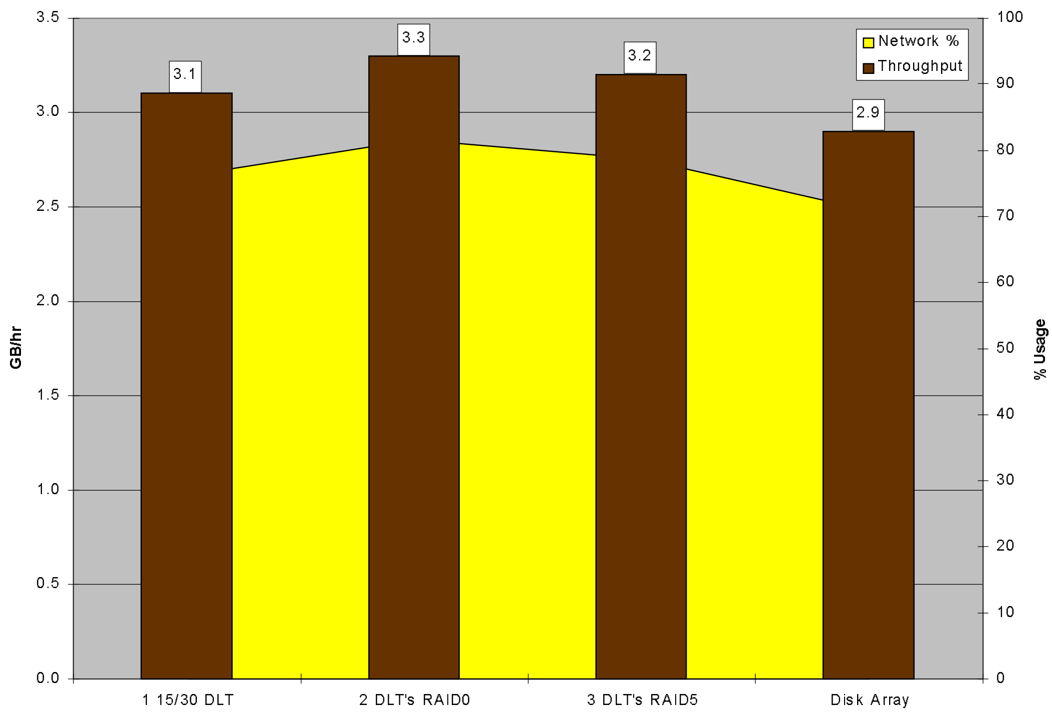
- Compaq NetFlex-3 and Netelligent network interface cards.
- Compaq Netelligent 10Base-T Repeater.
- Compaq Netelligent 100Base-TX Repeater.
- CAT 5 UTP cables (supports both 10 Mbs & 100 Mbs).

Both servers were cabled to the 10Base-T repeater for the 10 Mbit tests, then re-connected to the 100Base-T repeater for the 100 Mbit tests. The 100 Mbit technology chosen was 100Base-TX. Compaq supports other 100 Mbit Ethernet technology such as 100VG-Any LAN, but this was not tested. 100Base-TX supports full-duplex mode for bi-directional throughput up to 200 Mb/s. However, since most of the data transfer done here is unidirectional (from source server to destination server), benefits of full-duplex would be minimal and were not tested.

## Single Server Backups - 10BaseT Networks

Here we show the performance characteristics of backing up a database residing on one server to storage devices on another server, by transmitting a single data path across a 10Base-T network segment. Using ARCserve and its RAID option, the backup was done to a single 15/30-GB DLT tape drive as well as to an array of from 1 to 3 tape drives. SQL Server native dump was used to send data to a disk array.

**Chart 11 - Backup of One Server Across Single 10BaseT Segment**




The results obtained here are somewhat predictable. Given that the maximum transfer rate of a DLT tape drive (not counting compression) as well as a 10Base-T network are the same (1.25 MB/sec), one would expect to saturate the network wire with just a single DLT at the destination end. Furthermore, no increase in performance would be expected by striping multiple DLT's into a tape array using the ARCserve RAID Option. This is exactly what is shown in Chart-11, where the **network saturates at**

**over 75% average usage with just one DLT**, and performance gain with multiple DLT's is minimal. Of course using RAID-5 on the tape array does give the benefit of greater data protection.

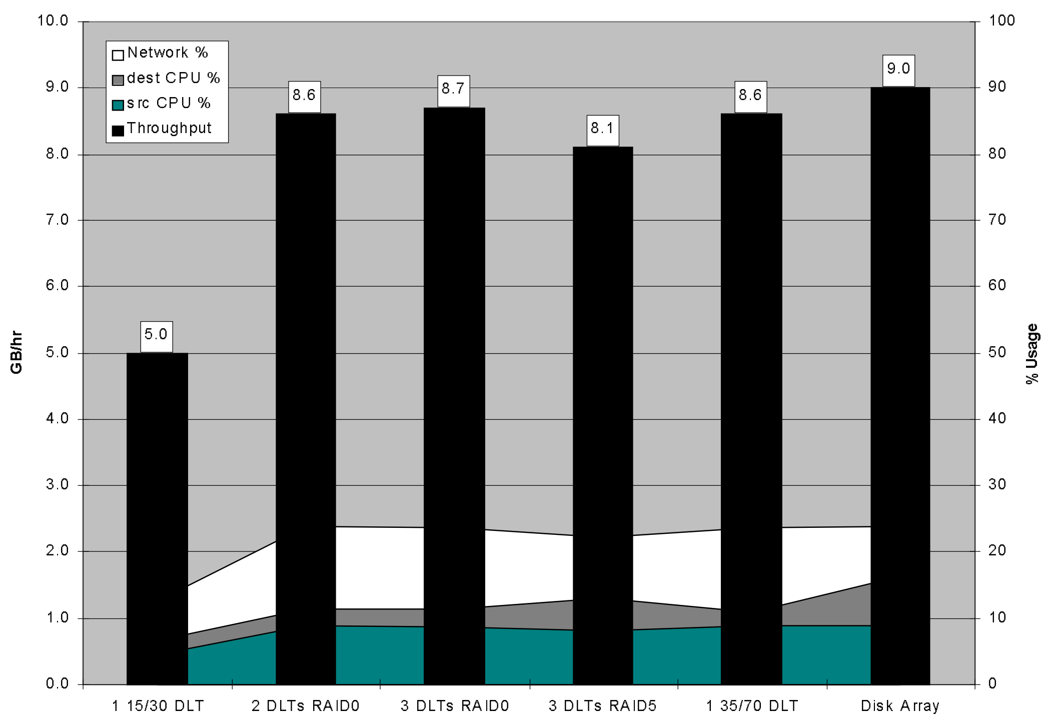
A dump to a disk array, although similarly constrained by the network, yielded slightly different performance than the tape arrays. This variance may be attributed to the difference in data transfer methods: the ARCserve database agent uses multiple named pipe streams to send data to the ARCserve application on the other end; SQL Server is sending a single data stream directly to the Windows NT file system.

Although not displayed, the CPU usage at both servers for all transfers was very low, at around 5%. Clearly the network is the bottleneck here, with plenty of system resources left at both servers for processing other jobs. Given the network saturation, such backup jobs should be scheduled during hours of low network use by other applications to avoid interference, or across a dedicated network link.

## Single Server Backups - 100BaseTX Networks

Here we perform the same tests as before, except that the 10Base-T repeater has been replaced with a 100Base-TX repeater. This makes the network capable of 100 Mbps transfers, which should theoretically be 10 times faster than the 10 Mbps transfers shown previously. The results obtained here however, do not follow this oversimplified prediction, as shown by the below chart:

**Chart 12 - Backup of One Server Across Single 100BaseTX Segment**




Comparing Chart-12 to Chart-9, the throughput seen while backing up data to a single 15/30-GB DLT drive over a 100 Mbit network is the same as the throughput for backing up to a local DLT drive, using ARCserve. A definite gain in throughput is also seen here by going from a single tape drive to an array (all tape arrays consisted of 15/30 DLT's). The performance of a 3 drive RAID-5 tape array is only slightly less than for a 2 drive RAID-0 array, which is as expected (see the discussion for Chart-9). No throughput gains are seen beyond two drives in RAID-0 however (8.6 GB/hr), and this same performance is seen with the single 35/70-GB DLT which can usually write data much faster. In fact, the performance for 100 Mbit is never more than 3x higher than for 10 Mbit, yet while monitoring the network utilization, it was never seen to exceed an average of 25%. The CPU usage at both the source and destination servers is also moderate, not exceeding 10% and 16% respectively.

This **lack of ability to saturate the 100 Mbit wire** could be attributed to a combination of things, but is mostly due to idle time on the wire between successive transmissions. The sending node can only transmit a certain number of bytes across the network before it must pause and wait for an acknowledgment (ACK) from the receiving node[72]. Idle periods result in an overall sustained throughput that is much lower than the instantaneous transfer rate between the two nodes. The positive side is that enough bandwidth remains on the wire for other network tasks to complete without very much interruption.

**Multiple Server Backups - 10BaseT Networks**

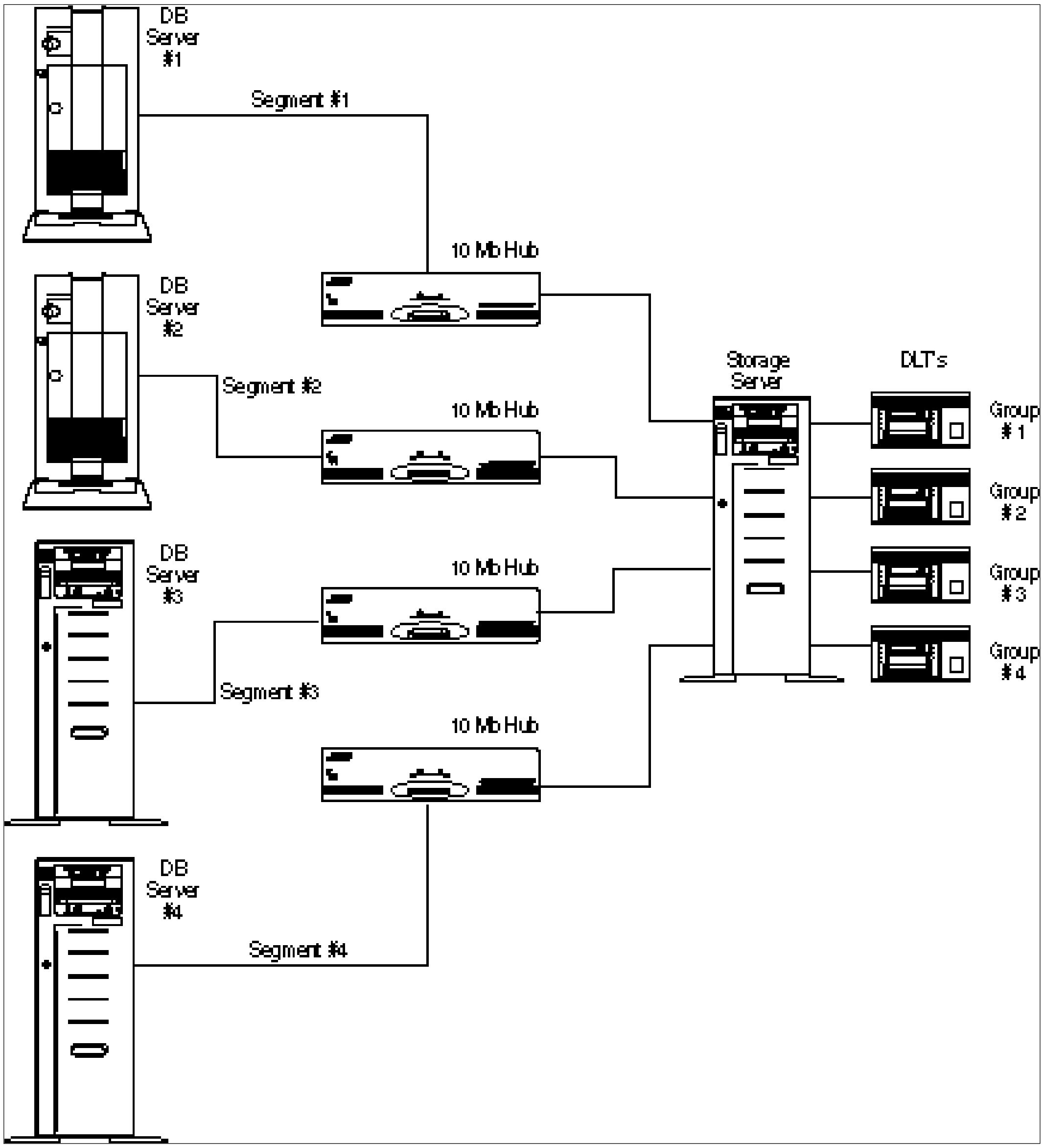
We now change to a concurrent, or parallel backup scheme, meaning that more than one database server is backed up at a time to a single storage server. This requires that the network layout be changed as well. Four database servers are placed on the network: two Proliant 5000 systems and two Proliant 2500 systems[73]. Each of these four systems is connected to a dedicated 10Base-T repeater (hub), and each repeater is connected to one of four network cards within the single storage server. The network layout is shown in Figure-1. This layout ensures that **a dedicated 10 Mbit path is available from each database server to the storage server**.

---

[72] The number of bytes transmitted without receiving an ACK may be adjusted by varying the size of the TCP receive window, through use of a TCP/IP parameter in the Windows NT registry (called **TcpWindowSize**). The default size (with ethernet) is 8760, meaning that 6 frames of 1460 bytes each may be sent consecutively. A larger receive window may improve performance with high delay or high bandwidth networks, although we did not notice much difference in our environment.

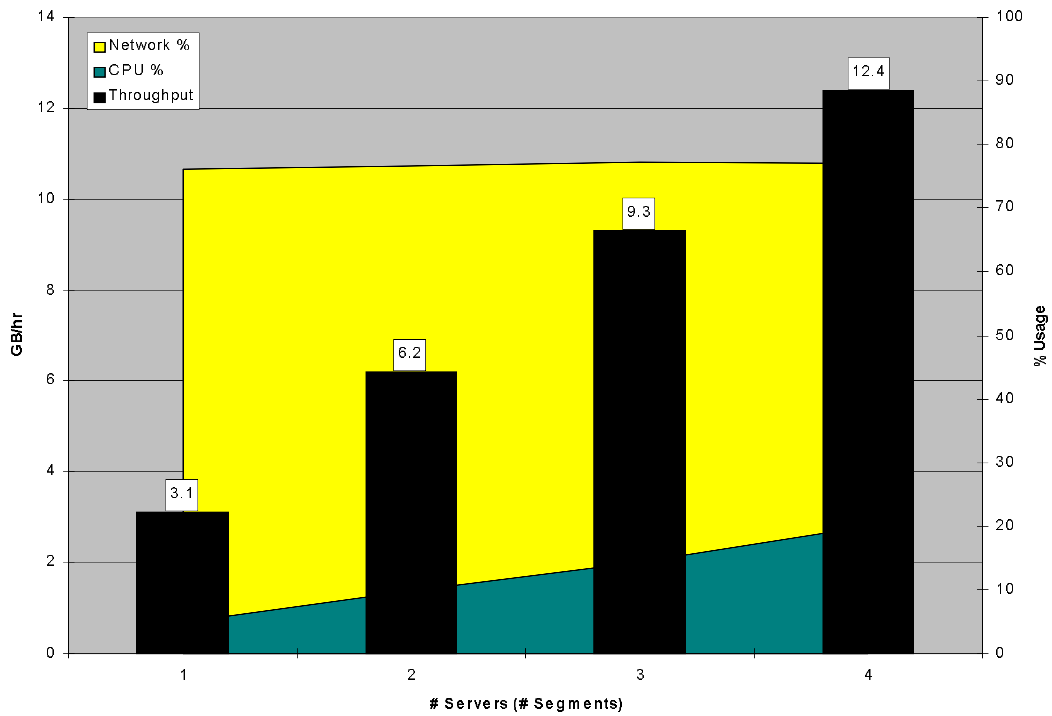
[73] See beginning of section for system specifics. Although the Proliant 5000 systems are more powerful than the Proliant 2500's, for the purposes of these tests they behave identically.

**Figure 1 – Network Layout: Backup of 4 Servers Across 4 Dedicated 10Base-T Segments**




At the storage server, four tape drive “groups” are set-up under ARCserve, one for each server to be backed up. Each drive group contains only a single 15/30-GB DLT drive, for it has already been shown that there is no performance benefit with more than one DLT per 10 Mbit segment. With this setup, backup jobs were initiated simultaneously on from 2 to 4 database servers. The following chart shows the throughput results as well as the CPU usage on the tape server. The network % figure shown is an average of the usage seen on each of the segments involved in the respective backup runs.

**Chart 13 - Concurrent Backup of Multiple Servers Across Multiple 10BaseT Segments**




As can be seen, **near perfect scalability was achieved in going from 1 to 4 concurrent backup jobs**. Whether backing up 1 server or 4 servers, the amount of time was the same, thus yielding 4x overall throughput increase at the destination server. CPU use at the storage server also increased linearly, but did not exceed a 20% average. Network utilization on each of the segments involved was about the same - roughly 76%. Thus, we can conclude that the simultaneous backup of servers across multiple segments is a good way to increase the productivity of your storage server and reduce overall backup time.

It should be noted that more than four such concurrent backup jobs could feasibly be performed with the addition of still more tape drives and NICs or dual-port NICs such as the Compaq Netelligent Dual 10/100TX[74].

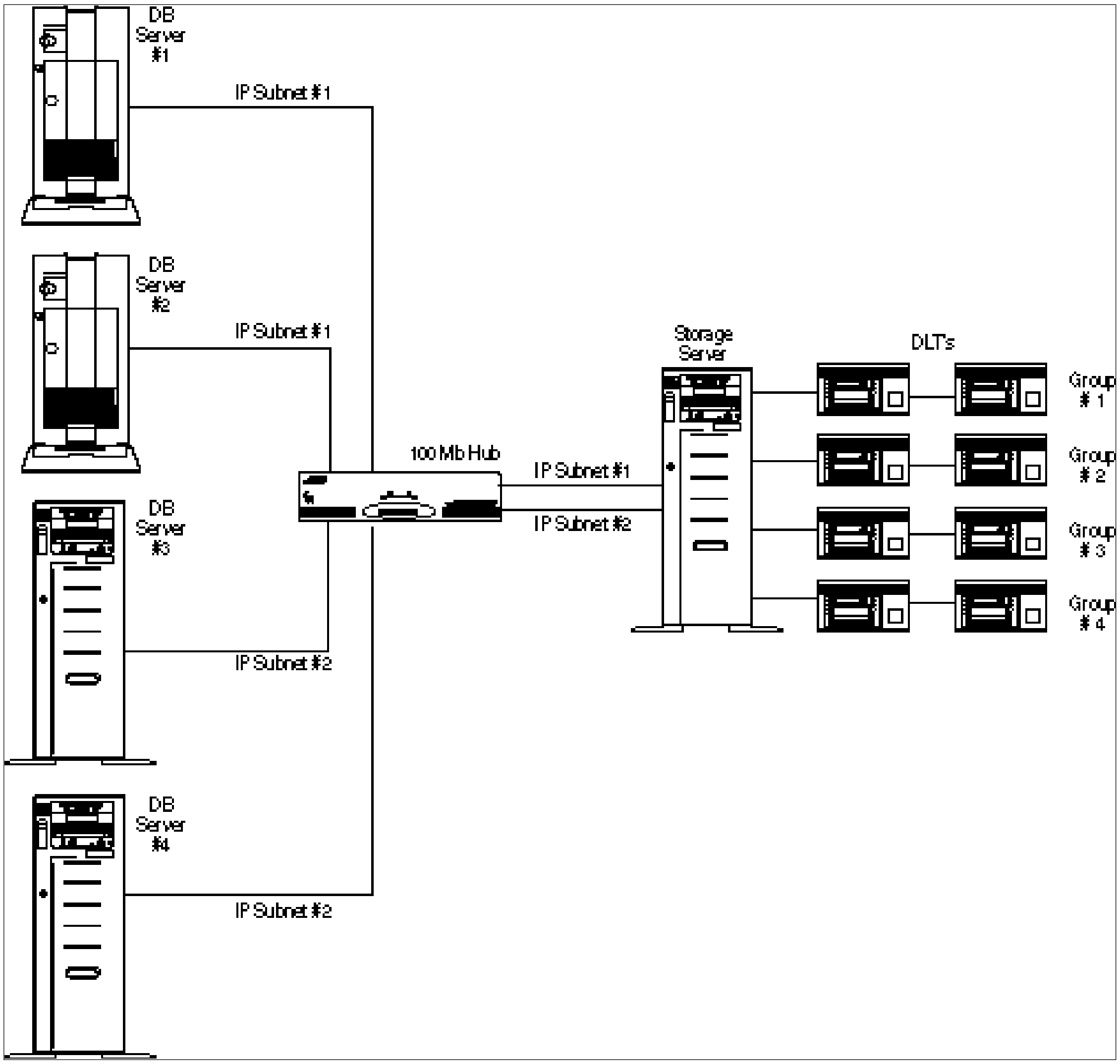
## Multiple Server Backups - 100BaseTX Networks

The concurrent backup scheme is continued here, with 100 Mbit technology. There is however, a key difference in the way the network is setup, versus the 10 Mbit tests. Earlier we saw that the 100Base-TX network wire had significant bandwidth remaining while performing a single remote backup. Thus, here **a single 100 Mbit 'segment' will be used to backup more than one server in parallel**. In our environment, this was accomplished by connecting the 4 database servers to a single 100Base-TX repeater (hub), and then connecting 2 of the network cards on the storage server to that same repeater. TCP/IP addressing was used to create two separate logical subnets within the single physical segment, so that 2 of the database systems could communicate with one of the NICs in the storage server and the remaining 2 database systems could communicate with the other NIC in the server. Thus we maintained a 2:1 ratio between systems to be backed up vs. network cards in the storage server, although all data transfers occur over the same wire. The reason for having more than one NIC on the

[74] Our configuration did not include performance testing with the Compaq Netelligent Dual 10/100TX PCI UTP controller.

storage server is to avoid creating a serialization point at the storage server's network interface while it processes incoming requests from multiple sources[75]. The network layout is shown in Figure 2.

**Figure 2 – Network Layout: Backup of 4 Servers Across 1 Shared 100Base-T Segment**



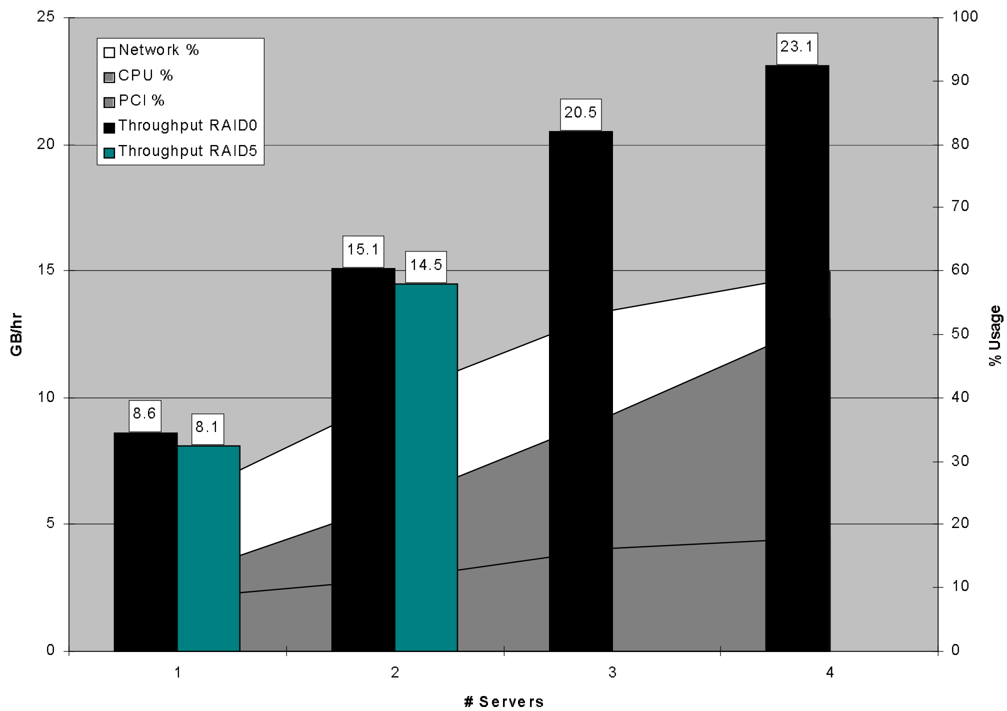

The storage server dedicated a tape drive group for each concurrent backup job using the ARCserve Manager, and the RAID Option was used to stripe the drives in each group. RAID-0 arrays of 2 drives each were configured for each concurrent backup job up to 4 jobs, and RAID-5 arrays of 3 drives were configured for up to 2 jobs. More than 2 RAID-5 arrays could not be configured as ARCserve supports only 8 drives and each RAID-5 array requires at least 3 drives. With this setup, backup jobs were initiated simultaneously on from 2 to 4 database servers. The below chart shows the throughput results, average CPU and PCI bus[76] usages on the tape server, and the average utilization of the network:

[75] Slower performance was observed when using only a single NIC on the storage server to receive data from up to 4 other servers.

[76] The PCI % usage shown in the chart was taken as an average of the load observed between the 2 peer-buses.

**Chart 14 - Concurrent Backup of Multiple Servers Across Single 100BaseTX Segment**




The benefits of this concurrent strategy are obvious, as **greater throughput is derived from the tape server and more of the network bandwidth is utilized**, for each additional database backed up from a remote system. Eventually, the overall throughput reaches 23.1 GB/hr with 4 simultaneous channels of data being transferred into the server and written out to tape, and 60% of the network bandwidth in use.

The performance increase however, is not as linear as what was observed during the concurrent 10Base-T tests. This is because all data here flows through a single point, and the CSMA/CD access method (used by 100BaseTX) leads to frame collisions when multiple nodes are transmitting on one segment. In the above tests, our collision rate went from 4% with 2 concurrent backups to over 21% with 4 concurrent backups. One could isolate the data transfers onto different segments and thereby reduce collisions by using a 100 Mbit switch in place of the repeater, by using 2 separate repeaters (one for each NIC on the server), or by using 4 separate repeaters (along with 4 NICs on the server) as we did for the 10 Mbit tests.

Note that Chart-14 does not include results for concurrent remote backups to 35/70-GB DLT drives. We would expect however, that such results would be very similar to the results for the RAID-0 configured 15/30-GB DLT drives (i.e: 4 groups of a single 35/70 DLT each could be substituted in place of the 4 groups of 2 15/30 DLT's in RAID-0), as they were in Chart-12. Remember that overall backup throughput is limited by the rate of the "slowest link in the chain" so to speak, and with remote backups that link is usually the network, so that a faster storage device does not always yield performance gains.

Finally, it should be mentioned that if a single segment is used for such concurrent backup jobs, then they should be scheduled 'after-hours' or done across a dedicated network link, so as to cause minimal interference with other network based applications. Furthermore, the ultimate performance of these or any other remote backup processes may vary based upon the specific physical layout of your network, especially in the case where a network bridge or routing device is configured in between the source and destination nodes[77].

[77] Bridges or routers placed in between the tape backup server and the other servers can significantly degrade network backup performance.

## Comparison of Methods for Online Backup

The DLT drives are Compaq's highest performing tape backup solution, and when configured as an array with the use of SQL Server 6.x striping or ARCserve RAID Option this performance is scaled even further. By adding enough of the drives, we have shown that we can attain backup throughput that is limited only by the ability of the disk subsystem to deliver the data. Of course disk subsystem data-transfer rates will continue to increase as more advanced SCSI and Fiber Channel technologies emerge, and these combined with high-end DLT drives such as the 35/70-GB DLT, will allow us to attain unprecedented backup throughput even beyond what is demonstrated in this paper. The DLT drives also provide us with the type of capacity sufficient for even the largest SQL Server databases: two DLT Tape Arrays (eight 15/30-GB DLT drives) can be combined into a single array that provides at least 120GB of storage space (no compression) on a single set of tapes, and with just one DLT Tape Array II (four 35/70-GB DLT drives) at least 140GB of space (no compression) is available. With the DLT 15-Cartridge Library storage capacity is extended to at least 225GB (these numbers are increased if compression is enabled).

The DLT hardware and media are rated for superior reliability, even under continuous operation. It is important to consider however, that as multiple tape drives are combined into a stripe set the Mean Time Between Failure (MTBF) of the tape array as a whole is compounded due to dependancy upon a greater number of devices (e.g: an array of 4 tape drives is 4 times more likely to fail than a single tape drive). Thus, fault tolerance for the tape drives or tape sets may become a consideration. Software fault tolerance is not however, available with SQL Server 6.x striped dumps. The Data Warehousing or IT department seeking a fault tolerant backup solution should certainly look at the ARCserve options, which are available directly from Compaq.

Using disk drives for local dump operations instead of tape drives may offer certain benefits in performance that can be useful for those who have a limited time interval in which to complete the backup of a large database (although one can match the performance of disk array dumps with a sufficient number of striped 35/70 DLT drives). However, dumps to disks are generally not practical in and of themselves, due to capacity limitations and because disks do not offer readily removable media that can be inexpensively archived. Rather, they should be used in conjunction with a tape solution, where the database can be dumped rapidly to disk and then the dump device files backed up to tape at leisure using a file-based backup utility. The advantage here is that interruptions to user transactions on the database caused by a rapid online dump would be relatively brief. And unlike a file-based backup of the database files themselves which would involve shutting down SQL Server, doing an off-line backup of the dump files requires no shutdown. Such an off-line backup should not cause significant performance degradation to normal system activity, as it is not a processor-intensive operation (provided software compression is not used), and I/O interference with normal database transactions during the backup can be minimized by providing a dedicated disk volume for the dump files[78]. If the user does find it practical to perform dumps to disk, a fault-tolerant capable RAID controller should be a consideration, keeping in mind the performance aspects of different RAID-level arrays.

Other SQL Server backup strategies involving temporary storage to disk include dumping to remote servers across the network. Multiple database servers could send their data to a single Backup server, from where the dump images could be transferred to tape (the only tape drives needed would exist on the backup server). Also, a Standby server can be set up which receives dump images (data and transaction log) to its disks from a Primary SQL Server, then immediately loads these images in order to maintain a "warm" copy of the production data in case the Primary server should fail.[79] The need for

[78] A dedicated volume for dump files (or even a dedicated controller) is recommended not only for performance reasons, but also for data isolation in case of hardware failure.

[79] Details on implementing a "warm" standby server are discussed in the *SQL Server Administrator's Companion*, Appendix E.

temporary disk storage can be removed by using Cheyenne ARCserve, which can 'pull' data from multiple database servers and sent directly to tape on the Backup server. Such remote backup jobs can be run in a sequential (back-to-back) fashion, or they can run concurrently if multiple 10-BaseT segments or a dedicated 100-BaseT segment are available.

The below table is intended to assist the customer in determining which database backup solution using DLT tape drives best meets their unique data archival needs, from a performance standpoint. The performance numbers presented represent striped backup of a single database on a Proliant 5000 server to multiple locally attached tape drives only, and were derived from previous sections (more accurate numbers can be obtained by referencing those sections). Many production environments have a limited time 'window' during which it is acceptable to perform backup operations, such as overnight or during periods of low user activity. If the customer knows the amount of data residing in his database (in GigaBytes), and the time available for backup (in hours), then the minimum throughput needed can be calculated (in GB/hr). The below table can then be referenced to determine which backup hardware / software configuration would best meet the throughput requirement.

**Table 2 - Number of DLT Drives Needed for Desired Throughput**

| Throughput Desired (GB/hr) | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|
| **SQL Server with 10/20 or 15/30 DLT** | 1 | 2 | 4 | 5 | 7 | 8 | - | - | - | - |
| **SQL Server with 35/70 DLT** | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 3 | 3 | 4 |
| **ARCserve RAID-0 with 10/20 or 15/30 DLT** | 1 | 2 | 3 | 4 | 5 | 7 | 8 | - | - | - |
| **ARCserve RAID-0 with 35/70 DLT** | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 4 | - | - |
| **ARCserve RAID-5 with 10/20 or 15/30 DLT** | 3 | 3 | 4 | 5 | 6 | 8 | - | - | - | - |
| **ARCserve RAID-5 with 35/70 DLT** | 3 | 3 | 3 | 3 | 3 | 3 | 4 | - | - | - |

Example: Suppose that an MIS department is planning a large, 40 GB SQL Server database (including transaction log) that will be heavily accessed by workers all day except for a brief period from 12 AM to 4 AM when no worker shift is active and only light, read-only activity needs to occur. The data administrators decide that it would be best to complete the entire archival of data to tape in about 2 hours, therefore their throughput requirements are a minimum 20 GB/hr. Looking at the above chart, we see that this throughput could be achieved with a number of different configurations. However, the MIS manager determines that a full backup of the database must be performed at the beginning of each week, with transaction log backups being performed every day thereafter to the same tape set (the tape set is rotated weekly). The daily log backups usually add up to another 40 GB total.

These additional requirements pose capacity considerations, as each tape set must be capable of holding around 80GB. If the department were to use 15/30-GB DLT tape drives, they would need a minimum of 5 such drives (assuming at least 6.25% compression). With five (5) 15/30 drives, the department could plausibly achieve a 20 GB/hr throughput if they use SQL Server's backup striping, and complete their backup job in around 2 hours. However, greater performance and about the same capacity could be achieved with fewer drives of the 35/70-GB variety. Therefore, a configuration using SQL Server 6.5 to archive the database to two (2) 35/70-GB DLT tape drives on a Proliant 6000[80] is chosen. By our estimates, this solution should be able to deliver a throughput of around 35 GB/hr and provide a storage capacity of at least 70 GB (minimum 12.5% compression would be needed on the data).

[80] Although the tests in this paper were done on a Proliant 5000 system, the same configurations on a Proliant 6000 would yield very similar results, since both systems are based on a similar (peer-PCI bus) architecture.

Note that the MIS manager has decided it is acceptable to restart or possibly lose backup jobs if a tape drive/cassette fails, and so ARCserve and its RAID option - along with their added costs - were not considered. Let's say that fault tolerance had been required: then an ARCserve RAID-5 array would most likely be implemented, and this could affect the choice in hardware. With RAID-5 a minimum of three (3) 35/70 DLT's would be required, providing 70 GB of capacity at just above 30 GB/hr. However, could achieve slightly greater capacity and performance with 6 of the 15/30-GB DLT drives, and at slightly less cost. Even five (5) 15/30 drives in RAID-5 might be sufficient for the job, with 60 GB of capacity (the data would need to compress 25%) at close to 20 GB/hr. However, because they want to allow for growth in the database size (which could increase both capacity and performance requirements), 6 drives would be the safer choice.

Finally, if the department had the entire night (say about 8 hours) to perform the database backup, then the DLT 15-Cartridge Library along with ARCserve (and its autochanger support) may have provided a better solution, allowing for unattended backups with even more capacity so that the data could be archived more frequently if needed. Backup to the tape library could also be done across a 100-BaseT network without affecting the completion time (performance would be about the same as doing it locally).

## *SQL Server Database Backup with User Activity*

This section will address considerations of performing on-line backups of a SQL Server database during user activity. Scenarios involving both the SQL Server native dump as well as the ARCserve assisted (database backup agent and RAID option)[81] dump will be presented.

**Dynamics of SQL Server Backup under User Activity**

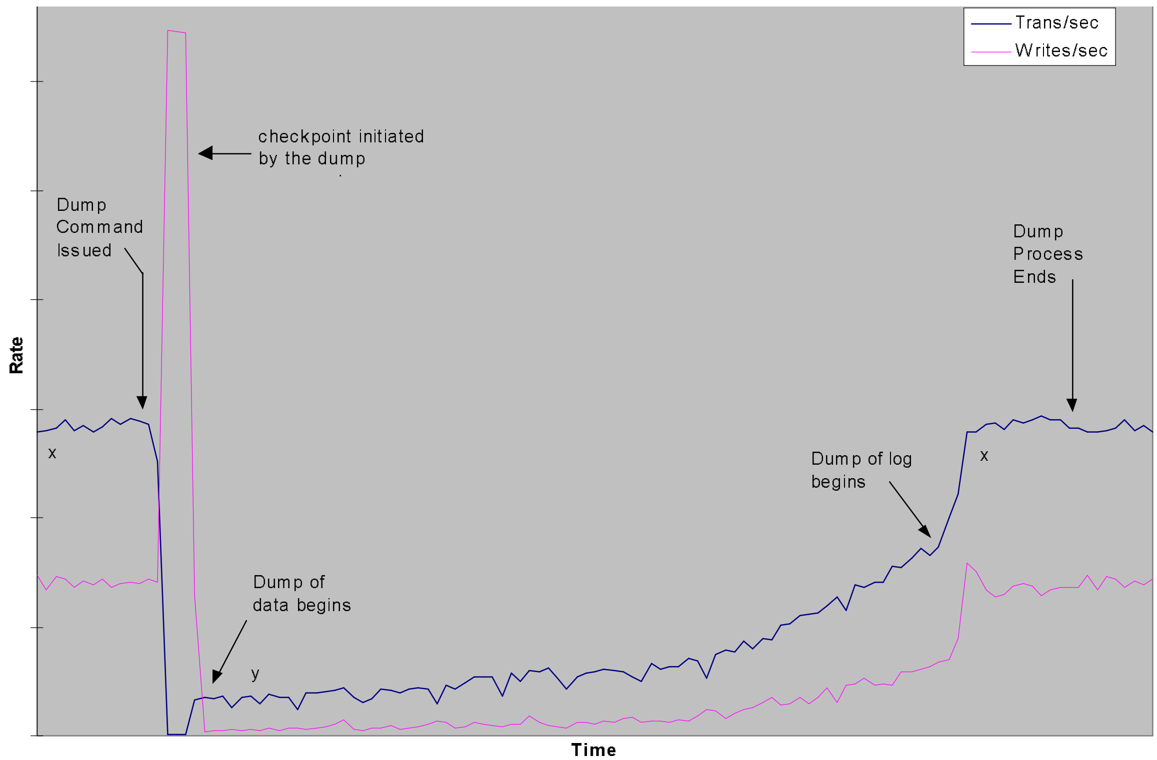
When you perform a dump of a database under user activity, several things happen. These are outlined below:

1) SQL Server is executing queries sent by the users at a throughput of *x*-transactions per second.
2) SQL Server receives a dump database command.
3) SQL Server issues an immediate checkpoint to flush all dirty pages to disk.
4) When the checkpoint finishes, SQL Server starts dumping the contents of the database, to the destination device (in the case of a tape drive, the tape is rewound first). SQL Server continues processing user queries (including updates); however, the throughput is reduced to *y*-transactions per second at the beginning of the dump.
5) If a user transaction causes an update to a page not yet backed up, SQL Server will back up the old value of this page prior to changing its contents and then will allow the transaction to complete. Any transactions updating a page already backed up will proceed immediately. Thus, SQL Server insures that the backup image is consistent with the state of the database at the time the dump command was issued.
6) When the database dump begins, most of the user transaction update (write) requests will be for data pages that have not yet been backed up. As the dump process progresses sequentially through the database, an increasing number of the write requests begin to fall on pages that are already backed up and can be modified immediately. This trend results in **a gradual increase in the user transaction performance as the dump proceeds.**
7) When SQL Server finishes dumping the data portion of the database, it begins dumping the transaction log. At this point the user transactions resume processing at the original throughput (*x*-transactions per second). This rate then continues as the log dump finishes and the dump procedure comes to an end.

Figure 3 represents this sequence graphically. Note the "performance curve" representative of the gradual increase in transaction rate as the dump proceeds. The exact form of this 'curve' will depend on the read-write ratio of your user transactions (reads are not affected by the dump). In our case the transaction rate went from about 10% of the original throughput to about 50% of the original throughput; the overall reads-to-writes ratio for our transactions was about 2-to-1.

[81] The ARCserve products were discussed earlier in the section: **Online Backup Considerations with ARCserve for Windows NT**.

**Figure 3 - SQL Server Backup with Database Activity**




Note that if a checkpoint command is issued during the database dump, it may not complete until near the end of the dump when most of the dirty pages that need to be written out have been backed up. Furthermore, write requests from the checkpoint process may delay user transaction writes thereby further affecting transaction performance. Automatic checkpointing can be disabled for the duration of the dump by setting the **recovery interval** parameter to a very high value, then resetting it immediately after the dump.

## Measuring Performance of SQL Server Backup under User Activity

In order to measure performance of SQL Server dumps under user activity, we engineered our environment in such a way, that we were able to control the level of load at the server via the introduction of a "think time" or a slight delay for our artificial users between submitting transactions.

We started by measuring maximum throughput in transactions per second, or 100% load at the server, by allowing users to submit transactions as fast as they could, with no induced delays. Next, we introduced delays to achieve 25%, 33% and 75% of the maximum transactions per second throughput rate (1/4$^{th}$, 1/3$^{rd}$, and 3/4$^{th}$ load at the server). Finally, we measured throughput (both user transactions per second throughput and GB/hour backup throughput) during a database dump to two Compaq 15/30-GB DLT drives, with the SQL Server native striped dump as well as the ARCserve assisted dump. During each of the phases we also noted changes in the transaction response times as would be experienced by the end-user terminals.

We defined two scenarios: 1) dump of an active database, and 2) dump of an inactive database. An active database would be one under a variable user activity, yielding a load at the server (we chose to test the 1/3$^{rd}$ load with this scenario). An inactive database would be one with no user activity, but another database in the system would be active at the same time, again yielding some load at the server (we chose to test the 1/4$^{th}$ and 3/4$^{th}$ loads with this scenario). Note that both of these situations are considered to be "online" backups, since in neither case is the database being backed up made inaccessible, although it is actively accessed only in the first scenario. The two environments do however, result in quite different performance dynamics.

The system that was used for this testing was a Proliant 5000 server with two (2) Pentium Pro/166 processors and 1 GB of RAM. The system had enough memory and spindles (disk drives) to allow for a maximum transactional throughput bound by the CPU's. Two 15/30-GB DLT drives were attached to the system's integrated Fast-Wide SCSI-2 controller. Windows NT 4.0 and SQL Server 6.5 were installed on the system and a large database was created and loaded.

The SQL Server **SMP Stat** and **Boost Priority** parameters were both deactivated (set to 0) for these tests. Enabling either of these parameters increases the priority of SQL Server threads relative to other process's threads under Windows NT. We have found that performing online backups with user activity can cause CPU utilization to rise significantly, and increasing SQL Server priority can increase the risk of a system 'lock-up' should all CPU's reach 100% usage. Setting SMP Stat to 0 or 'n-1' (where n = # of CPU's) is especially important when using a third party application (such as ARCserve) to assist in the backup, or else the SQL Server threads can starve the backup application and halt the backup process[82]. Finally, we recommend that an SMP system with at least two CPU's be used if online backups with user activity are to be done.

**Online Backup of Active Database**

Table 3 and Table 4 display the results of dumping a database that is under user activity using SQL Server and ARCserve, respectively. The database was under enough activity to generate a load that is 1/3rd (33%) of the system's maximum[83].

**Table 3 - Online Dump of Active Database with SQL Server**

| **Transactions Before Dump** | | | **Transactions During Dump** | | | **Dump** |
|---|---|---|---|---|---|---|
| **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Throughput:** |
| **No User Transactions** | n/a | ~ 0% | No User Transactions | n/a | ~ 6% | 9.1 GB/hour |
| **33% of Maximum** | .12 sec | ~32% | 4% to 16% of Maximum | 1.42 sec to 0.65 sec | ~64% | 8.9 GB/hour |

**Table 4 - Online Dump of Active Database with ARCserve**

| **Transactions Before Dump** | | | **Transactions During Dump** | | | **Dump** |
|---|---|---|---|---|---|---|
| **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Throughput:** |
| **No User Transactions** | n/a | ~ 0% | No User Transactions | n/a | ~60% | 10.0 GB/hour |
| **33% of Maximum** | .12 sec | ~32% | 3% to 12% of Maximum | 1.91 sec to 0.87 sec | ~99% | 10.0 GB/hour |

[82] Setting SMP Stat to 0 still allows SQL Server threads to use all CPU's present in the system, but guarantees that one CPU will be available for other processes by moving SQL threads off that CPU if necessary. Refer to the *SQL Server Administrator's Companion* for more information on this parameter.

[83] We do not recommend attempting a backup of a database that is under a much heavier user read/write load.

When dumping an active database with SQL Server, the user transaction throughput significantly decreases during the dump operation: From 33% of the maximum throughput before the dump, it fell to 4% of the max at the start of the dump then gradually rose to about 16% of the max near the end of the dump. Average transaction response times also increased by about 12 times (start of dump) to 5 times (end of dump). As explained earlier, any transaction updating a page that has not already been backed up is placed on an internal queue, where it waits until SQL Server checks this queue, determines which page needs to be dumped next in order to "release" this transaction, dumps this page, and then allows the transaction to complete. This is believed to be the bottleneck seen in our testing.

When using ARCserve to back up an online database, the ARCserve SQL Server Backup Agent actually interfaces with the native SQL Server dump process to retrieve the data. Thus, we can expect the affect on user transactions to be more or less the same as it was with the SQL Server dump alone. Table 4 however, when compared to Table 3, shows us that the transaction rate and response times were affected more by the ARCserve assisted dump than by the SQL dump. The reason for this is because the ARCserve dump has encountered another bottleneck: the system CPU's, which are saturated at 99%. As discussed earlier in the paper, the ARCserve backup process demands more CPU time than the native SQL Server backup, thereby having a potentially greater affect on other system activity. When we added two more CPU's to our system and re-ran the ARCserve test, the affect on user transactions was about the same as with the SQL dump (these results are not shown in Table 4).

Finally, as for the throughput of the dump operation, it fell slightly from 9.1 GB/hr (maximum dump throughput achieved with 2 DLT drives with no user activity) to 8.9 GB/hr during the native SQL Server dump with user activity, and was unaffected during the ARCserve dump with user activity. In general though, **it is the user transaction performance and not so much the dump performance that is affected** during a backup of an active database[84].

**Online Backup of Inactive Database**

For this scenario, a second database was created on the system. This database was placed on its own disk array separate from the first database, so that I/O from activity on one database would not interfere with processes on the other database. While user transactions were simulated on the first database, the dump process was initiated on the second database.

Table 5 and Table 6 display the results of dumping a database while another database on the same SQL Server is under user activity, using SQL Server and ARCserve, respectively. The database was placed under variable activity to generate a load that was first 1/4$^{th}$ (25%) of the system's maximum, and then a load that was 3/4$^{th}$ (75%) of the system's maximum, to simulate both moderate and high use conditions.

**Table 5 - Online Dump of Inactive Database with SQL Server**

| Transactions Before Dump | | | Transactions During Dump | | | Dump |
|---|---|---|---|---|---|---|
| **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Throughput:** |
| **No User Transactions** | n/a | ~ 0% | No User Transactions | n/a | ~ 6% | 9.1 GB/hour |
| **25% of Maximum** | .12 sec | ~26% | 25% of Maximum | 0.12 sec | ~32% | 9.1 GB/hour |

[84] This is true for an OLTP-type environment with numerous read/write transactions. A Decision Support-type environment involving large read-only queries may behave differently since reads are not affected by the backup process.

| | | | | | | |
|---|---|---|---|---|---|---|
| **75% of Maximum** | .15 sec | ~70% | 75% of Maximum | 0.17 sec | ~80% | 6.6 GB/hour |

**Table 6 - Online Dump of Inactive Database with ARCserve**

| Transactions Before Dump | | | Transactions During Dump | | | Dump |
|---|---|---|---|---|---|---|
| **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Transaction Rate:** | **Response Time:** | **CPU Usage:** | **Throughput:** |
| **No User Transactions** | n/a | ~ 0% | No User Transactions | n/a | ~ 60% | 10.0 GB/hour |
| **25% of Maximum** | .12 sec | ~26% | 25% of Maximum | 0.13 sec | ~80% | 10.0 GB/hour |
| **75% of Maximum** | .15 sec | ~70% | 44% of Maximum | 2.90 sec | ~99% | 9.3 GB/hour |

When dumping an inactive database with SQL Server, the user transactions per second throughput (to the active database) and the average transaction response times do not change considerably under either of the load conditions. The fact that we have isolated our two databases on separate disk arrays helps to produce this situation **-** I/O requests from the backup job do not interfere with user transaction I/O.

The dump throughput, while it remained unaffected during the moderate (25%) user load condition, dropped from 9.1 GB/hr (maximum dump throughput achieved with 2 DLT drives with no user activity) to 6.6 GB/hr under the high (75%) load. Here, the dump threads compete with other SQL Server threads processing user transactions against another database. Since no artificial bottleneck is imposed on any of the user transactions, SQL Server schedules them equally with the threads performing the dump operation. SQL Server can have as many worker threads as there are users, up to a certain limit, which, once reached, causes user connections to share a worker thread. Naturally, the two dump threads receive only a fraction of the execution time, resulting in low backup throughput when a high user load is present. With a lighter user load, backup throughput does not suffer because there is enough CPU time for all threads (in Table 5, CPU usage is at 32% during the dump with light load, vs. 80% for the heavy load). Note that the results may be somewhat different if you are dumping to a greater number of backup devices, in which case there will be a greater number of backup threads.

When doing the ARCserve assisted dump, the results for the moderate user load condition are the same (neither user transaction throughput nor backup throughput are affected), but the results for the heavy load condition are a bit different: The user transaction rate dropped from 75% of the maximum throughput to 44% of max, and the average response times increased about 19 times. The backup throughput on the other hand, only dropped from 10.0 to 9.3 GB/hr.

Here, the SQL Server user transaction threads must compete with ARCserve application threads which are external to SQL Server. Since we have the **SMP Stat** parameter set to 0, Windows NT will move SQL Server threads off of one of the CPU's to make room for other applications' threads when the CPU's near saturation (Table 6 shows CPU usage at 99% in this case). When SQL threads are deprived of some time on this CPU, user transactions are affected. If we had set SMP Stat to 2 or **-**1, the user transactions would not have been affected, but the ARCserve backup process would have suffered or even halted entirely (see discussion on SMP Stat earlier in this section).

From our results we can make the following general conclusions about backup of an inactive database during activity on another database: Neither the backup throughput nor the transaction throughput will be affected as long as there is adequate CPU time to service all threads and I/O is separated. **If CPU usage becomes high however, the backup throughput will suffer** (with the SQL Server dump process), **or the transaction rate will suffer** along with a reduction in backup throughput (with the ARCserve backup agent).

## *Online Transaction Log Backup Performance*

Like the database itself, the transaction log can be backed up with the database still online using SQL Server dump functionality. A transaction log can be backed up as part of the database, or by itself if it resides on a separate database device. The section in the first chapter entitled **Transaction Log vs. Database Archiving** discusses the uses of transaction log-only backups.

### Characteristics of Transaction Log Dumps

There are a number of different ways in which a database's transaction log can be dumped, and the performance can differ depending upon which of these methods is used. The commands that can be issued to SQL Server to dump a transaction log are listed below. For more details concerning these commands or when to use them, consult the *SQL Server Administrator's Companion.*

- **DUMP DATABASE:** The transaction log is dumped as part of the full database dump. After transferring to tape the data pages in all of the other tables, SQL Server will then begin reading pages from the *syslogs* table and transferring them to tape as part of the dump. When done in this manner, the transaction log is read sequentially using large size (up to 16KB) asynchronous requests just like the rest of the database, and is not truncated.
- **DUMP TRANSACTION:** The transaction log is dumped by itself. As the log table is dumped, it is simultaneously purged of committed transactions, or truncated. The truncation process forces the need to do additional, smaller size I/O (including 2KB writes) to the log volume. Although the SQL Server Read Ahead Manager[85] is invoked to assist with the truncate, the additional I/O required limits the overall performance of the dump transaction operation.
- **DUMP TRANSACTION WITH NO_TRUNCATE:** The transaction log is dumped by itself, but is not purged of committed transactions. This command eliminates the need to perform additional I/O required by a truncate. However, all log pages are read using single page (2KB size) synchronous requests, therefore limiting the performance.
- **DUMP TRANSACTION WITH TRUNCATE_ONLY** or **DUMP TRANSACTION WITH NO_LOG:** Both of these commands will purge the log of its committed transactions without creating a backup image. Since no data is actually sent to a dump device, the performance of this operation can be faster than for other dump transaction operations, but is still limited by the need to do some smaller size I/O.

These characteristics can apply when using third party online backup tools as well, such as Cheyenne's ARCserve for Windows NT and its Database Backup Agent, which initiate a DUMP command within SQL Server[86].

### Log Dump Throughput

To determine the maximum throughput possible for each of the above operations, we issued the commands using a NULL device instead of a tape or disk drive.

*When dumping the transaction log as part of the database*, it is possible to see throughput as high as that of the database dump itself, as it is essentially part of the same process. However, as the transaction log usually is stored on a separate, often smaller disk volume, the **rate at which it can be read may differ from the rate at which the rest of the database is read**. For high-speed backups of the transaction log, we recommend a log volume consisting of at least **4** Fast-Wide SCSI-2 drives in a RAID-1 configuration, or **3** such drives in a RAID-5 configuration. The below table shows some

---

[85] The Read Ahead Manager is a mechanism which can issue separate thread(s) to pre-fetch data during an operation involving sequential reads.

[86] The ARCserve database agent performs SQL Server transaction log dumps similar to the way in which it does database dumps. See section entitled **Online Backup Considerations with ARCserve** for details.

common transaction log volumes (fault tolerance and number of drives) that may be implemented on an array controller, along with the approximate throughput they can be expected to deliver when performing read operations during a SQL Server dump:

**Table 7 - Max Dump Throughput for Small Arrays**

| RAID Level | Spindles | Throughput |
|---|---|---|
| 1 | 2 | 12.5 GB/hr |
| 1 | 4 | 20 GB/hr |
| 1 | 6 | 25 GB/hr |
| 5 | 3 | 25 GB/hr |
| 5 | 4 | 30 GB/hr |

*When dumping the transaction log by itself*, the need to also truncate the committed transactions seems to **limit the operation to a maximum throughput** of about **15** GB/hr. *When dumping the transaction log by itself and avoiding truncates*, the small, single-page reads used limit the operation to about **12** GB/hr**.** Increasing the number of spindles in the log volume does **not** yield increases in these numbers.

The rate at which a transaction log dump can write data to a single tape drive can also differ from the rate seen for the database dump, depending on the amount of compression the drive is able to achieve on the transaction log vs. compression on the data. Transaction logs generally lend themselves well to compression, although this can depend on the nature of the transactions recorded in the log (e.g.: transactions which repeatedly modify the same or similar rows will yield a higher compression rate on the log). In our environment, a transaction log dump to one DLT drive yielded throughput higher than for a database dump (**6.5** GB/hr vs. **5.0** GB/hr). Unexpectedly however, this number did not increase **-** and in fact went down **-** when a striped dump was done to multiple DLT drives! The SQL Server **Dump Transaction process does not seem to be able to keep the tape drives streaming for a striped operation**. The results were similarly low when selecting the option to avoid log truncates. The reason for this is because the DUMP TRAN process uses only a single thread to write to all of the output devices, even when doing a striped operation. When using an ARCserve agent to perform the transaction log dump to multiple drives however, this performance limitation was bypassed; as ARCserve uses its own, multiple threads to write out the data.

The below table summarizes some of the results encountered with transaction log dumps:

**Table 8 - Max Transaction Log Dump Throughput with Various Commands**

| Command | Software | To Device(s) | Throughput |
|---|---|---|---|
| **w/ DUMP DATABASE** | SQL Server | NULL | Max[87] |
| | SQL Server | 1 15/30 DLT | 7.2 GB/hr |
| | SQL Server | 1 35/70 DLT | 20 GB/hr |
| | SQL Server | 8 15/30 DLT's | 29 GB/hr |
| | ARCserve | 8 15/30 DLT's | 34 GB/hr |

[87] Throughput depends on disk configuration used to hold transaction log volume.

| | | | |
|---|---|---|---|
| **DUMP TRANSACTION** | SQL Server | NULL | 15 GB/hr |
| | SQL Server | 1 15/30 DLT | 6.5 GB/hr |
| | SQL Server | 1 35/70 DLT | 15 GB/hr |
| | SQL Server | 8 15/30 DLT's | 6.1 GB/hr |
| | ARCserve | 8 15/30 DLT's | 14.8 GB/hr |
| **NO TRUNCATE option** | SQL Server | NULL | 12 GB/hr |
| | SQL Server | 1 15/30 DLT | 5.1 GB/hr |
| | SQL Server | 1 35/70 DLT | 10.6 GB/hr |
| | SQL Server | 8 15/30 DLT's | 5.4 GB/hr |
| | ARCserve | 8 15/30 DLT's | 11.3 GB/hr |

### Estimating Total Backup Time

When performing a full database dump, it is important to include the transaction log as part of your calculation in estimating the overall time needed to backup the database. For most environments, it will be safe to use the following "rules of thumb" to help in this estimation:

- If both your database and log dump will be limited by the throughput of the system (tape drives, network link, etc.), then the size of the transaction log can be added to the size of the database, and the total size divided by the throughput of the system:

  **[db_size + log_size] / [system_throughput] = backup_time**

  The exception here would be when the transaction log space yields compression significantly different from the data space.

- If the transaction log dump will be limited by the read-throughput of the log volume, then the rate at which the log can be dumped must be considered separately:

  **[db_size / system_throughput] + [log_size / log_volume_throughput] = backup_time**

The system throughput (in Gb/hr) for various configurations can be obtained from the charts in previous sections. Log volume throughput limitations (in GB/hr) are discussed above. The size of your data and log spaces (in GB) should be obtained using one of the methods discussed in 'Online Backup Considerations with SQL Server Dump'. Time (in hrs) can then be estimated using the above formulae.

Example: Need to do a full database backup of a 27.3 GB database (used space) with a 850 MB transaction log (used space), using SQL Server dump. The database server is a Proliant-5000 with a Smart-2/P array controller. The database is constructed across 10 disk drives configured in RAID-5, spanning both SCSIports of the Smart-2/P (5 per port). 2 more drives on the Smart-2/P comprise the volume for the transaction log, configured in RAID-1. The system has available 4 DLT drives which are connected to 2 SCSI-2/P controllers. The amount of time needed to complete this backup needs to be estimated.

We must compare the maximum throughput of the disk drive volumes to the maximum throughput of the tape array. Chart-4 shows that 1 Smart-2/P can deliver about 50 GB/hr (if there are at least 4 drives per SCSIport), so the data volume can be read very fast. The log volume is much smaller however, capable of only about 12.5 GB/hr (see Table-7). Chart-6 shows that it is possible to achieve around

16.5 GB/hr with 4 DLT drives, which exceeds the rate of the log volume. Therefore the backup time must be calculated as:

[27.3 / 16.5] + [.85 / 12.5] = 1.72 hours, or about 1 hour & 43 minutes.

If only 2 DLT's were to be used to back up the database, a system throughput of 9.1 GB/hr would then be the limitation (Chart-6) for both log and data, simplifying the calculation:

[27.3 + .85] / 9.1 = 3.1 hours, or about 3 hours & 6 minutes.

In the second case however, it is possible that additional time may be required to change the tapes in the drives, since the total space required may exceed the capacity of two 10/20-GB cartridges (depending on the compression achieved).

Keep in mind that all such calculations yield only rough estimates of the required time, and should be used conservatively. Customers should perform their own tests to measure actual time, and in a mission critical environment should allow for extra time to incorporate any unforeseen performance degradation.

## *Database Recovery*

The goal of this section is to supplement information on recovery processes found in sources such as the *Microsoft SQL Server System Administrator's Companion*. The Administrator's Companion provides detailed information on different recovery scenarios, such as recovering from media failure, recreating lost devices, recreating and reloading lost databases, restoring and reloading the *master* database, etc., along with examples of each. This section provides additional considerations and a performance analysis of the recovery process.

### On-Line Recovery Considerations

Following are some functional considerations you should be aware of when loading a database after a failure:

- Make sure you create a database of the same or greater size than your original database. You will not be able to load a database into a smaller frame structure, even if only a portion of the database is used.
- If you are reloading the database at another server, the code page and sort order must match with the original installation. To move data between servers with different code pages and/or sort orders, use the BCP utility or SQL Transfer Manager.
- Create the database in the same fashion as the original one, with respect to its physical characteristics, such as device allocation, log space allocation, segment allocation, etc. The best method to ensure identical physical characteristics is to save the SQL scripts used to initially create your database(s), and rerun them prior to loading the database(s).
- The database will be loaded in the same order that it was created, with respect to device allocation.
- When recovering a database from a database dump and a series of subsequent transaction log dumps, all dump images must be loaded in sequence and successfully. If, for example, you load from a database dump, and another user updates a record in this database before you have a chance to load the next transaction log dump, the transaction log load will fail. When this happens, you are forced to start over and reload from the last database dump. Evidently, the update transaction occurring between this load will be lost. To prevent this situation from happening, start SQL Server in a single user mode.
- During a restore operation, the damaged database must be dropped, recreated and the most recent copy of the database dump is then loaded from the backup media. If the database is not damaged, you can reload the backup over the old database. SQL Server locks the database being restored for the duration of the operation. However, all other databases remain open for access.

- After restoring the data pages during a load, SQL Server will reinitialize any unused pages remaining in the database. Depending on what percentage of the database remains unused, this process can consume a significant amount of the overall load time, and generates much I/O in the form of 16KB writes to the data device.
- When re-creating a database for the purpose of restoring it from a dump, use the FOR LOAD option in the CREATE DATABASE statement (new for SQL Server 6.0), in order to skip the initialization of data pages. Since unused data pages are initialized as part of the load process anyway, skipping this step during database creation can save time and I/O. Also, SQL Server 6.x will now perform the I/O to create the database in large (up to 64KB size) units, thus improving the speed.
- During a load, SQL Server reads data from the dump device (disk or tape) in 64KB chunks[88], and into the in-memory buffer. From there, they are written out to the disk in 16KB extents (new for SQL Server 6.0), which are units of 8 contiguous data pages. Thus, roughly 4 writes to the data volume should occur for every read request from the dump volume or tape drive.
- If you are loading one database at a time from a single device, performance of the backup device will generally be the limiting factor. If you are loading multiple databases concurrently, or are loading a database from multiple, striped devices, then the performance of 16KB writes to the data volume may become the limiting factor. This is largely dependent upon the type of fault tolerance implemented on the data array, and the number of spindles in the data array.
- Performance during loads will be negatively affected by implementing a fault tolerance of RAID-5 (striping with parity) on the database volume. This is because when writing data back to a RAID-5 disk array, 4 physical I/Os (2 reads and 2 writes) must normally occur for every logical write request so that the parity information may be re-created. The impact of this is partially offset by the Smart-2 controller’s ability to detect a sequential write stream and thereby minimize the amount of parity information re-created.

Other important considerations can be found in the Microsoft SQL Server manuals and the *Microsoft SQL Server, Backup and Recovery Guidelines* document, available from Microsoft.

## Database Load Performance

The purpose of this section is to provide information on the database load performance, which can then be compared to database dump performance from previous sections. Measuring the performance of a database load is more complicated than for a dump. This is because, as mentioned above, the load process actually consists of two phases: 1. Reading the data from the backup media while restoring it to the database on disk, and 2. Reinitializing all the pages in the database which remain unfilled. Thus, the performance of both of these I/O intensive operations must be measured separately in order to gauge the overall database load throughput for a system, and thereby accurately predict load time.

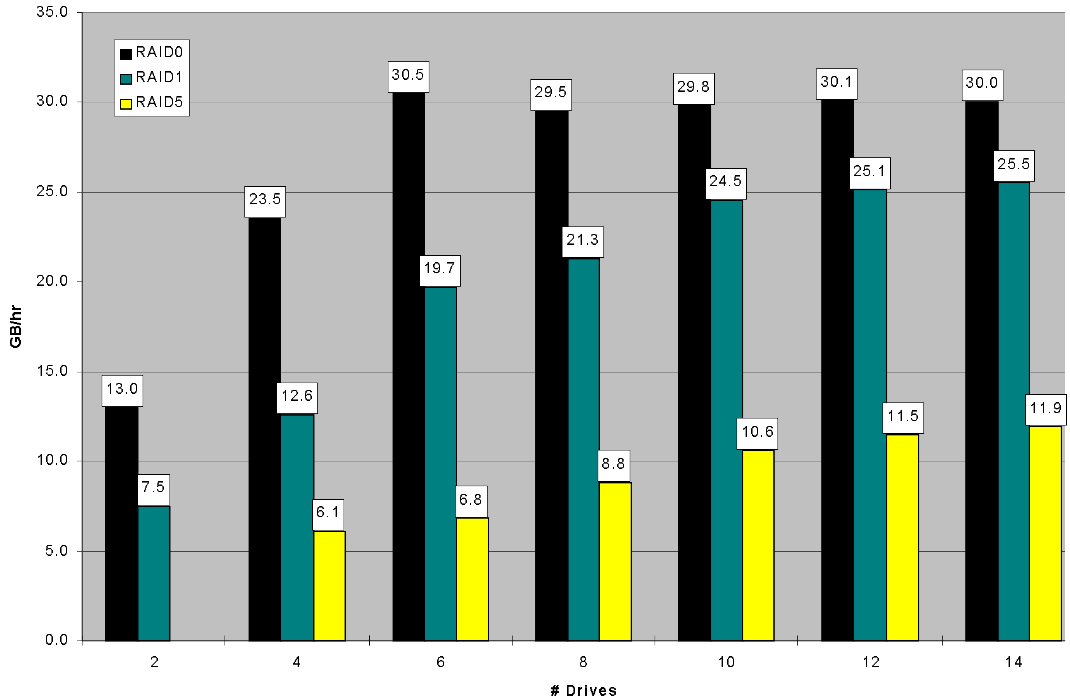
Furthermore, database load performance is more dependent upon the nature of the disk volume housing the database, than is database dump performance - especially when restoring from multiple, striped devices. Therefore, the type of data array in the system is an important consideration during both phases of the load process, as it may very well determine the performance bottleneck.

For our analysis of load performance we use two charts: one for each phase of the load process. For each set of tests we also vary the number of spindles (disk drives) and RAID level used on the data array. All loads were done from a stripe set of eight 15/30-GB DLT tape drives using the native SQL Server load functionality. The system was a Proliant 5000 with a single Smart-2 SCSI array controller.

[88] The 64KB read from a tape device occurs in blocks based on the corresponding tape block size.

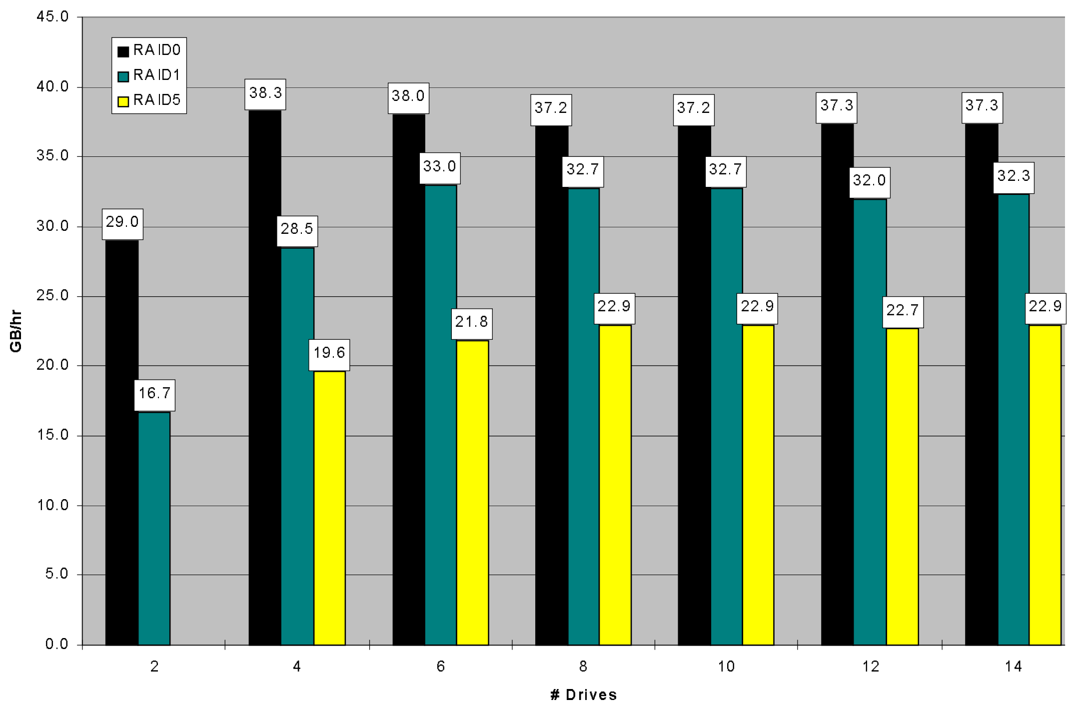
These results do not include any activity needed to search, replace or rewind[89] the tapes, which would naturally prolong the restore process.

**Chart 15 - Load Database: Data Restore Performance based on Disk Array Type**




[89] Rewind of the tapes between the restore and reinit phases of the load can be prevented by using the NOUNLOAD option.

**Chart 16 - Load Database: Page Reinitialization Performance based on Disk Array Type**




From the above results it is at once apparent that the type of data disk array does indeed have a huge impact on both phases of the database load. The 8 tape drive array employed had a data transfer rate of about 30 GB/hr; faster than most of the numbers seen in Chart-15. Thus, in all cases except for the RAID-0 tests, performance of the restore phase was limited by the data array, and generally improved when spindles were added to the array. Performance of the RAID-1 array did not improve much beyond 10 drives (~ 25GB/hr) because at this point the Smart-2 SCSI controller became saturated[90].

The performance of the RAID-5 array during the restore phase is especially bad. This is because the I/O profile to the array during this process consists entirely of random writes, and forces additional I/O's for parity. The I/O is random because 8 tape drives are being read almost simultaneously by 8 different threads, and although the data stream coming from each of the tape drives is sequential, the combination of all the threads writing out the data becomes randomized. The load performance to a slow data array from multiple DLT drives can also suffer if the tape drives are not kept streaming[91]. Restore performance with SQL Server, especially to a RAID-5 array, is therefore one area where **using more tape drives yields less benefit than would restoring from a single high speed device**. Restore from a single 35/70-GB DLT to a 12 spindle RAID-5 array proceeded at the speed of the drive (about 19 GB/hr), faster than with 8 15/30-GB DLT's (11.5 GB/hr). Interestingly, using the ARCserve Backup Agent for SQL Server with 8 DLT drives yielded significantly better restore performance to the RAID-5 array: 16.6 GB/hr with 14 spindles (versus 11.9 GB/hr with SQL Server only). This is because the data is read off the 8 tape drives by ARCserve and passed to SQL Server, which then uses less threads to write the data to the disks yielding a less random write profile. Restore performance to the 14 spindle RAID-0 and RAID-1 arrays was not affected by using ARCserve. ARCserve results are not displayed in the above charts.

[90] Drive mirroring actually writes twice the amount of data to its disks. So at a 25 GB/hr throughput, the controller was really writing 50 GB/hr of data, or about 7 MB/sec to each of its SCSI ports.

[91] As when writing out data during a backup operation, tape drives during a restore perform best when data requests are coming in continuously, so that they do not have to stop, re-position the tape, then continue reading.

The page reinitialization phase of the load (chart 16) is generally faster with all types of arrays than the initial phase. Although both processes employ 16 KB writes to the data array, the reinitialization is a purely sequential operation (which translates to less head seek times on the disk drives), and does not need to fetch data from the tape drives. As with most sequential operations, performance does not continue to increase beyond a certain number of spindles in the array.

**Estimating Database Load Time**

Using charts 15 and 16, we can estimate the total amount of time that is needed to load a database from a database dump. Since a full database dump image consists of both the database and its transaction log, we must account for the time consumed by 4 separate process, listed in order of operation as follows: **1**. The restore data pages in the database, **2**. The restore of pages in the *syslogs* table (the transaction log), **3**. Reinitializaiton of unused pages in the database, and **4**. Reinitialization of unused log pages. The charts above apply to both data and log restores (meaning log restoration done as part of the LOAD DATABASE command). However, since the array used to house the transaction log may differ from that used to hold the database, it will be necessary to use one of three possible methods:

- ❑ If both the data and log parts of the load will be limited by the throughput of the system (tape drives, network link, etc.), then the size of the transaction log can be added to the size of the database, and the total size divided by the throughput of the system to find the restore times. Only the reinitialization rates are considered separately:

  **[(db_size + log_size) / system_throughput] + [db_free_space / db_reinit_rate] + [log_free_space / log_reinit_rate] = load_time**

  The exception here would be when the transaction log space yields compression significantly different from the data space.

- ❑ If the data load will be limited by the throughput of the system, and the log load will be limited by the write-throughput of the log volume, then the rate at which the log can be restored must be considered separately. The reinitialization rates are also considered separately.

  **[db_size / system_throughput] + [log_size / log_volume_throughput] + [db_free_space / db_reinit_rate] + [log_free_space / log_reinit_rate] = load_time**

- ❑ If both the data and log parts of the load will be limited by the write-throughputs of their respective volumes, then the rate at which each can be restored must be considered independantly. The reinitialization rates are also considered separately.

  **[db_size / db_volume_throughput] + [log_size / log_volume_throughput] + [db_free_space / db_reinit_rate] + [log_free_space / log_reinit_rate] = load_time**

The system throughput (in Gb/hr) for various configurations can be obtained from the charts in previous sections, as the actual performance of the storage interface (tape drives or network link) is generally the same while storing data as when retrieving it (the exception is disk storage) . Data or Log volume throughput limitations (in GB/hr) are shown in Chart-15. Reinitialization rates are shown in Chart-16. The size of your data and log spaces as well as the unused amounts in each (all in GB) should be obtained using one of the methods discussed in 'Online Backup Considerations with SQL Server Dump'. Time (in hrs) can then be estimated using the above formulae. Note that if the database and transaction log reside on the same disk volume, then these formulae can be simplified by considering the log as part of the database.

Example: A full database backup was performed on a database with 27.3 GB of used data space, and 850 MB of used transaction log space, using SQL Server dump[92]. This backup set now needs to be

[92] Using SQL Enterprise Manager it is possible to determine the total size of the backup image on tape, but it is not possible to determine how much of that space was for data and how much for the log. Thus, it is advisable that the user label his or her tapes with this information.

loaded into an empty database with 50 GB of total data space and 2 GB of total log space. The database server is a Proliant-5000 with a Smart-2/P array controller. The database is constructed across 10 disk drives configured in RAID-5, spanning both SCSIports of the Smart-2/P (5 per port). 2 more drives on the Smart-2/P comprise the volume for the transaction log, configured in RAID-1. The system has available 4 DLT drives which are connected to 2 SCSI-2/P controllers (the same tape configuration was used to back up the system). The amount of time needed to complete this load needs to be estimated.

We must compare the maximum throughput of the disk drive volumes to the maximum throughput of the tape array. Chart-15 shows that a 10 drive RAID-5 array can write around 10.6 GB/hr (from multiple restore threads[93]), so the data volume cannot restore very fast. The log volume is slow also, capable of only about 7.5 GB/hr. Chart-6 shows that it is possible to achieve around 16.5 GB/hr with 4 DLT drives, which exceeds the rate of both data and log volumes. Therefore the load time must be calculated as:

[27.3 / 10.6] + [.85 / 7.5] + [(50 - 27.3) / 22.9] + [(2 - .85) / 16.7] = 3.75 hours, or about 3 hours & 45 minutes.

The reinitialization rates of 22.9 GB/hr and 16.7 GB/hr for the data and log volumes respectively, were obtained from Chart-16. We can compare this time to the 1.72 hour time that was needed to dump the database on the same system (see example in presented in 'Estimating Total Backup Time'). The database load time is over 2 times greater than the dump time, due to the restore performance of the disk volumes and the need to reinitialize unused pages.

Now lets say that the drive volumes for this system were configured as 10 drives in RAID-1 for the data, and 4 drives in RAID-1 for the log, so that the restore throughputs increase to 24.5 and 12.6 GB/hr respectively. Both the database and log restore are now limited by the tape subsystem, so that the time calculation becomes:

[(27.3 + .85) / 16.5] + [(50 - 27.3) / 32.7] + [(2 - .85) / 28.5] = 2.44 hours, or about 2 hours & 26 minutes.

The reinitialization rates of 32.7 GB/hr and 28.5 GB/hr for the data and log volumes respectively, were again obtained from Chart-16.

Keep in mind that all such calculations yield only rough estimates of the required time, and should be used conservatively. Figuring out the total time needed to load a database is a complex procedure, and may be quite specific to individual systems. Customers should perform their own tests to measure actual time, and in a mission critical environment should allow for extra time to incorporate any unforeseen performance degradation.

### Transaction Log Load Performance

Depending upon the type of backup strategy which you have implemented, recovery of a database may not be complete after restoration of data and log pages from a full database dump. There may be successive transaction log-only dumps which then need to be loaded, in the order that they were performed. For a discussion of backup strategy involving transaction log dumps, as well as details on the actual transaction-logging process, please see the chapter entitled **Data Protection Concepts**. This section will attempt to give an idea of the amount of time required to load a transaction log dump.

In the previous section, the performance of loading a transaction log as part of a full database load was considered. That process consists of simply restoring the dumped log pages from media into the database's *syslogs* table. Loading a transaction log-only image (i.e: one performed with a DUMP TRANSACTION command) however, consists of restoring the log as well as re-executing all of the data changes recorded in the log - a process known as "recovering the database" from the transaction log. Also called "applying" the transaction log, this process rolls forward all committed transactions

[93] This number is a 'worse case' since the chart was based upon 8 restore threads and the example uses only 4.

that were recorded (but not previously applied), and rolls back any transactions that were uncommitted at the time of the log dump.

During the first phase of the transaction log load, SQL Server seems to restore the log in sections, by reading the existing log to obtain buffer space in memory, reading from the backup media and writing to the buffer space (from where it is written out to the log file by the log writer), then pausing to read the existing log again, etc. This causes a ‘stop-go’ behavior on the backup media if it is a tape device. The performance of this phase is mainly limited by the type of disk volume upon which the log resides.

The second phase of the transaction log load, recovery and application of the log, is what consumes the vast majority of the time. While sequentially reading the newly restored log volume, SQL Server performs random I/O (2KB reads and writes) to the data volume in order to re-enact the changes recorded in the log. While the performance of this phase is somewhat determined by the type of disk volume upon which the database resides, it will not proceed at above a certain rate.

**Sample Transaction Log Load**

There is no way to predict exactly how much time is required to apply the all of the changes in a transaction log of a certain size. This is because the nature of the transactions recorded in the log can determine how long it will take to apply the log, not just the size of the log. Things which can affect the “log space -to- apply time” ratio include:

- ❑ The type of Data Manipulation Language (DML) statements recorded in the log. For example, UPDATES can use up more log-space than INSERTS or DELETES, since they may require that both a ‘before’ and ‘after’ snapshot of the updated row be recorded.
- ❑ The width of the data rows affected by the DML statements. Modifications to wider rows will generally take up more space in the log. This may cause any log replay rate figured in GB/hour (log space / apply time) to appear higher with such rows.
- ❑ The number of indexes affected by the DML statements, as well as the width of these indexes.

Given the above variables, the best we can do is to present a sample system and transaction log image, then relay the amount of time needed to load that image, in hopes of providing an idea of the load time.

Our transaction log contained 450 MB of statements generated from simulated OLTP-type[94] insert and update transactions. The system was a Proliant 5000 with 2 P5/166 CPU’s, 4 15/30-GB DLT drives, and a disk configuration involving two Smart-2/P controllers. The first controller had 12 disks configured in RAID-0 (for performance) holding the database. The second controller had 4 disks configured in RAID-1 holding the log. The time required to load this transaction log was as follows:

- Restore of Transaction Log: 5.5 minutes. Approximately 4.9 GB/hr.
- Application of Transaction Log: 2 hours, 50 minutes. Approximately .044 GB/hr.

When the data volume of 12 disks was reconfigured as RAID-5:

- Application of Transaction Log: 3 hours, 22 minutes. Approximately .037 GB/hr.

During application of the log, 115,406 transactions were rolled forward, and 1,196 transactions were rolled back. The average transaction consisted of 11 updates and 12 inserts, so the log contained around 116,602 * 23 = 2,681,846 DML statements. Incidentally, the amount of time that it took to actually generate the 450 MB log with a peak transaction load on the server, was **less** than the time needed to apply it during the load:

- RAID 0 data array: 43 minutes.

[94] On-Line Transacting Processing. A type of database environment characterized by many, relatively short transactions that generate a large number of random reads/writes to data and index objects spread throughout the database.

- RAID 5 data array: 1 hour, 32 minutes.

This will not necessarily be the case with transaction profiles containing a much higher ratio of read (non-logged) vs. write (DML) statements, or with a system under light usage.

Although transaction logs dumps take considerably less time and space to perform than full database dumps, using multiple such dumps to rebuild a lost database can greatly prolong the overall database recovery time. This point should be considered carefully when planning a database backup & recovery strategy.

### *Other Performance Considerations*

There exists a number of additional factors that can all influence performance of backup and restore operations. These factors include the following:

- Rewinding tapes
- Changing tapes
- Searching for the end of the backup set, when applicable (appending new data)
- Writing additional data onto tape, such as registry and/or catalog information
- Cataloging of tapes
- Backing up across the network when the network utilization is high (available network bandwidth)
- User activity on the database.
- Diagnostics and replacement of the failed hardware
- Creating and/or formatting of lost partitions, if necessary
- Re-creation of the database devices and the database itself, if necessary.

## Conclusion

The intention of this document is to communicate relevant information on backup and recovery technology available with Microsoft SQL Server for Windows NT and Compaq server products. We have presented different methods for database backup, and compared their functionality and performance (see the section entitled **Comparison of Methods for Online Backup**). Let us now conclude by summarizing the key points presented in this document.

1. Make sure your data is adequately protected by appropriate disk fault tolerance as well as a sound archiving strategy. Make sure you test this strategy thoroughly.
2. Determine the type of backup strategy: off-line or on-line. If you can afford shutting SQL Server down to back up your transaction log(s) and database(s) within an acceptable time interval, investigate the options available with an off-line backup. Otherwise, implement an on-line backup most suitable for your environment. Each method has its own benefits and limitations, and a careful analysis of your requirements will help you determine the best method. The performance of off-line backups was not considered in this paper.
3. Online backups can be performed with the SQL Server native database ‘dump’ capability, or with a third party application that provides a database backup ‘agent’. Cheyenne ARCserve 6.0 for Windows NT provides a Backup Agent in addition to other options.
4. Use data compression to increase the tape cartridge’s capacity and backup throughput. All compression methods used with online backups will be hardware (not application) based. The nature of your data will determine the effectiveness of data compression, and will directly impact your backup throughput.
5. For the highest throughput capability during a backup, you should build your database across a single, large array consisting of multiple disk drives. The best way to implement such an array is through use of a hardware RAID controller such as the Compaq Smart-2 SCSI Array controller.
6. You can increase your backup throughput by using multiple storage devices along with SQL Server striped database dumps or the ARCserve RAID option. The ARCserve RAID option can also provide fault tolerance for your backup sets.
7. You will be able to increase your overall throughput doing multiple backups simultaneously. You will achieve the highest throughput improvements and greatest tape drive scalability if you can preserve sequential I/O. Preserve sequential I/O whenever possible, by backing up only one database per logical volume at a time (assuming each logical volume is on its own drive array).
8. Dumping data to a disk array can also yield very high backup throughput. Data dumped to disk can later be archived onto tape using an off-line backup utility.
9. Remote backups offer a way to utilize a single Backup Server to archive data from multiple sources. SQL Server can dump a database to a remote disk share. ARCserve provides a way to do remote database backups directly to tape.
10. Dedicated networks and high-speed networks, such as 100Base-T, can greatly improve remote backup throughput. If using 10Base-T, consider having a separate network segment for every database server that you will be backing up concurrently.
11. The DLT Tape Library can be used with ARCserve to provide unattended backups and maximum storage capacity, although performance will be limited to that of a single drive. Tape striping can be used to increase capacity as well, therefore allowing for unattended backups in the case where all your data fits on a single ‘stripe set’.
12. When dumping a database under user activity, the user-transaction rate will be affected more than the dump throughput. When dumping a database while another database on the same system is under user activity, it is important to ensure adequate CPU power or else the backup throuhput will suffer and possibly the user-transaction rate as well (if using the ARCserve agent).

13. When estimating the amount of time needed to complete a full database dump, don't forget to account for the size of the transaction log. If archiving the transaction log separately, the performance of the log dump will usually be slower than that of a database dump.

14. The performance of a database load (or restore) can be less than that of a dump (backup), depending on the type of disk array you are restoring to; RAID-5 arrays can especially slow-down a striped load. The restore process is followed by a reinitialization of unused pages which further delays the operation. Your database load may need to be suceeded by one or more transaction log loads, which involve replaying much of the information contained in the log. You should plan for the **total** time needed to recover your data in case of a failure.

We are planning on delivering more documents on related subjects in the future. We strive to provide you with the best technical information in a timely fashion. Please take a minute to fill out the *User Registration/Evaluation Form* on the next page and return it to us at your earliest convenience. This will help us tailor the future information products to your needs, and will enable us to make future revisions of this document and related new information products available to you.

# Reference Material

## Software Vendors

### <u>SQL Server for Windows NT</u>

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
1-206-
http://www.microsoft.com/

### <u>ARCserve for Windows NT</u>

Cheyenne Software
3 Expressway Plaza
Roslyn Heights, NY 11577
1-516-465-4000
http://www.cheyenne.com/

## Publications

### <u>Compaq Computer Corporation</u>

Compaq Computer Corporation
20555 SH 249
Houston, Texas 77070
Main (281) 370-0670
http://www.compaq.com/

***Configuring Compaq RAID Technology for Database Servers***, published in February 1994 (Document Number 267A/0294)

***Compaq Backup and Recovery Of Microsoft SQL Server for Windows NT,* First Edition,** published in October 1994

***Configuration and Tuning of Microsoft SQL Server 6.5 for Windows NT on Compaq Servers***, **Third Edition,** published in June 1996 (Document Number 415A/0696)

# User Registration/Evaluation Form

Please fill out and return to us this registration/evaluation form to help us keep you up to date with future revisions of this document and related new information products. Your effort will help us improve the quality of the future information products.

**Name:** ______________________________
**Title:** ______________________________
**Company:** ______________________________
**Address:** ______________________________
______________________________
______________________________
**Phone Number:** ______________________________

Please evaluate the quality of this document:

| | Excellent | | | | Poor |
|---|---|---|---|---|---|
| **Technical Accuracy** | 1 | 2 | 3 | 4 | 5 |
| **Organization** | 1 | 2 | 3 | 4 | 5 |
| **Clarity** | 1 | 2 | 3 | 4 | 5 |
| **Completeness** | 1 | 2 | 3 | 4 | 5 |
| **Timeliness** | 1 | 2 | 3 | 4 | 5 |

Please indicate the type of environment you have at your site:

| Operating Systems | RDBMS | Processing Type |
|---|---|---|
| ❑ SCO UNIX | ❑ Microsoft SQL Server | ❑ On-Line Transaction Processing |
| ❑ Microsoft Windows NT | ❑ Sybase System 10 | ❑ Decision Support |
| ❑ IBM OS/2 | ❑ Oracle7 | ❑ Batch Processing |
| ❑ Novell NetWare | ❑ Other: | ❑ Other: |
| ❑ UnixWare | | |
| ❑ Other: | | |

Please indicate the type of information you would like us to provide in the future:

| Topic | Operating Systems | RDBMS |
|---|---|---|
| ❑ Configuration and Tuning | ❑ Microsoft Windows NT | ❑ Microsoft SQL Server |
| ❑ Capacity Planning | ❑ Novell NetWare | ❑ Sybase System 10 |
| ❑ Integration Information | ❑ IBM OS/2 | ❑ Oracle7 |
| ❑ Competitive Analysis | ❑ UnixWare | ❑ Other: |
| ❑ Systems Management | ❑ SCO UNIX | |
| ❑ Other: | ❑ Other: | |

**Additional Comments:**

**Return to:**
Database Engineering
Compaq Computer Corporation
MailCode 090803
20555 S.H. 249
Houston, Texas 77070