Logging Subsystem Performance: Model and Evaluation

Thomas K. Clark
Portland State University

Recommended Citation

10.15760/etd.6608

This Thesis is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. For more information, please contact pdxscholar@pdx.edu.
Thesis Approval

The abstract and thesis of Thomas K. Clark for the Master of Science degree in Computer Science were presented October 21, 1994 and accepted by the thesis committee and the department.

COMMITTEE APPROVALS:

Leonard Shapiro, Chair

Jingke Li

Robert H. Gerber
Informix Software, Inc.

Tavor Smith
Representative of the Office of Graduate Studies

DEPARTMENT APPROVAL:

Warren Harrison, Chair
Department of Computer Science

**************************************************************

ACCEPTED FOR PORTLAND STATE UNIVERSITY BY THE LIBRARY

by ________________________________ on 19 December 1994
Abstract


Title: Logging Subsystem Performance: Model and Evaluation

Transaction logging is an integral part of ensuring proper transformation of data from one state to another in modern data management. Because of this, the throughput of the logging subsystem can be critical to the throughput of an application. The purpose of this research is to break the log bottleneck at minimum cost.

We first present a model for evaluating a logging subsystem, where a logging subsystem is made up of a log device, a log backup device, and the interconnect algorithm between the two, which we term the log backup method. Included in the logging model is a set of criteria for evaluating a logging subsystem and a system for weighting the criteria in order to facilitate comparisons of two logging subsystem configurations to determine the better of the two.

We then present an evaluation of each of the pieces of the logging subsystem in order to increase the bandwidth of both the log device and log backup device, while selecting the best log backup method, at minimum cost. We show that the use of striping and RAID is the best alternative for increasing log device bandwidth. Along with our discussion of RAID, we introduce a new RAID algorithm that is designed to overcome the performance problems of small writes in a RAID log. In order to increase the effective bandwidth of the log backup device, we suggest the use of inexpensive magnetic tape drives and striping in the log backup device, where the bandwidth of the log backup device is increased to the point that it matches the bandwidth of the log device.
For the log backup interconnect algorithm, we present the novel
approach of backing up the log synchronously, where the log backup
device is essentially a mirror of the log device, as well as evaluating
other log backup interconnect algorithms.

Finally, we present a discussion of a prototype implementation of
some of the ideas in the thesis. The prototype was implemented in a
commercial database system, using a beta version of INFORMIX-OnLine
Dynamic Server™ version 6.0.
Logging Subsystem
Performance:
Model and Evaluation

Thomas K. Clark

Thesis for Master of Science in Computer Science
Portland State University
1994
Acknowledgments

I would like to thank my advisor, Dr. Len Shapiro, for his guidance and direction in focusing the content of this thesis and ensuring its scholarship. I would also like to thank the many people in research and industry for their insights and industry perspective on the ideas contained in the thesis, especially Jim Cray and the members of the Informix R&D facility in Portland, OR. Special thanks also goes to Informix Software, Inc., for allowing their software to be used in the prototype implementation. Thanks also to the members of the thesis committee for their time and effort in reviewing the contents of this thesis.

Finally, I would like to thank my wife Diane for her tremendous patience and support during the many long hours involved in this work.
Contents

Acknowledgments .................................................. 11

1 Introduction ..................................................... 1

2 The Logging Model and Previous Work ........................... 4
  2.1 Logging Methods ............................................. 5
  2.2 Hardware .................................................... 9
    2.2.1 Log Devices ........................................... 10
    2.2.2 Log Backup Devices .................................. 15
  2.3 Log Backup Methods ........................................ 19
  2.4 Evaluation Criteria ......................................... 24
    2.4.1 Loss of Data ........................................... 26
    2.4.2 Maximum Average Speed ................................ 27
    2.4.3 Maximum Burst Speed .................................. 28
    2.4.4 Cost/Performance ...................................... 29
    2.4.5 Soft Crash Recovery ................................... 29
    2.4.6 Hard Crash Recovery ................................... 31
    2.4.7 Special Hardware ....................................... 34
    2.4.8 Special Software ....................................... 35
    2.4.9 Weighting the Criteria ................................ 35
  2.5 Assumptions .................................................. 36

3 Log Device Bandwidth .......................................... 39
  3.1 The Parallel Logging Solution ............................... 39
  3.2 The Striping Solution ....................................... 42
    3.2.1 Reliable Striping - RAID ............................. 44
    3.2.2 Small Writes in a RAID Log: Append-Only RAID ....... 46

4 Log Backup Device Bandwidth .................................. 51
  4.1 Synchronous Backup: Striped Tape Beats Single Tape ...... 52
  4.2 Asynchronous Backup: Striped Tape Beats Single Tape ...... 55
  4.3 Offline Backup: Striped Tape Beats Single Tape .......... 58
  4.4 Staggered Allocation Backup: Striped Tape Beats Single Tape 59
  4.5 Conclusions .................................................. 61

5 The Interconnect - Evaluation of Log Backup Methods .......... 62
  5.1 Low Bandwidth ............................................... 63
5.1.1 Synchronous Interconnect Beats
Asynchronous Interconnect ........................ 63
5.1.2 Synchronous Interconnect Beats
Offline Interconnect ............................... 67
5.2 Medium Bandwidth .................................. 69
5.2.1 Synchronous Interconnect Beats
Asynchronous Interconnect ......................... 70
5.2.2 Synchronous Interconnect Beats
Offline Interconnect ............................... 73
5.3 High Bandwidth .................................... 76
5.3.1 Synchronous Interconnect Beats
Asynchronous Interconnect ......................... 77
5.3.2 Synchronous Interconnect Beats
Offline Interconnect ............................... 80
5.4 Conclusions ......................................... 82
6 Results from an Implemented Prototype .......... 84
6.1 The Implementation .................................. 84
6.2 The Benchmark ..................................... 87
6.3 The Results ......................................... 88
7 Conclusion ............................................. 94
Bibliography ........................................... 96
# List of Tables

2.1 Summary of Evaluation Criteria .................................................. 26  
2.2 Strategies for Hard Crash Recovery Without Redundancy .............. 34  
2.3 Log Criteria Weights ................................................................. 36  
4.1 Configuration Comparisons for Log Backup Device Bandwidth ........ 51  
4.2 Synchronous Interconnect Performance Per Dollar ..................... 53  
4.3 Synchronous Backup: Single Tape vs. Striped Tape .................... 55  
4.4 Asynchronous Backup: Single Tape vs. Striped Tape ................. 57  
4.5 Offline Backup: Single Tape vs. Striped Tape ......................... 59  
4.6 Staggered Allocation Backup: Single Tape vs. Striped Tape ........ 61  
5.1 Low Bandwidth: Synchronous vs. Asynchronous Cost .................. 64  
5.2 Low Bandwidth: Synchronous vs. Asynchronous Results .............. 66  
5.3 Low Bandwidth: Synchronous vs. Offline Cost ............................ 67  
5.4 Low Bandwidth: Synchronous vs. Offline Results ..................... 69  
5.5 Medium Bandwidth: Synchronous vs. Asynchronous Cost ............ 70  
5.6 Medium Bandwidth: Synchronous vs. Asynchronous Results .......... 72  
5.7 Medium Bandwidth: Synchronous vs. Offline Cost .................... 73  
5.8 Medium Bandwidth: Synchronous vs. Offline Results ............... 76  
5.9 High Bandwidth: Synchronous vs. Asynchronous Cost ............... 78  
5.10 High Bandwidth: Synchronous vs. Asynchronous Results .......... 80  
5.11 High Bandwidth: Synchronous vs. Offline Cost ...................... 81  
5.12 High Bandwidth: Synchronous vs. Offline Results .................. 82
## List of Figures

2.1 Overview of Logging Model ........................................... 5  
2.2 Phases of ARIES Restart Recovery ................................. 8  
2.3 Asynchronous Log Backup Method .................................. 20  
2.4 Synchronous Log Backup Method .................................... 21  
2.5 Offline Log Backup Method ......................................... 22  
2.6 Staggered Allocation Log Backup Method .......................... 23  
6.1 Offline and Synchronous Interconnect Results ................... 90  
6.2 Asynchronous Interconnect Results ................................. 92  
6.3 Asynchronous Interval Performance ............................... 93
Modern data management makes use of the transaction concept in order to transform data from one state to another in an orderly manner. A transaction is defined as a transformation of the data that maintains the four well known properties of

- Atomicity - actions are all completed or have no effect.
- Consistency - leaves the data in a valid state.
- Isolation - events in a transaction are hidden from other concurrent transactions.
- Durability - committed results of transactions survive subsequent failures [Gray81a].

The implementation of these so-called ACID properties is normally done through the use of some type of locking mechanism in conjunction with the keeping of a transaction log. The transaction log keeps track of the operations on the data, providing enough information to allow incomplete or failed transactions to be rolled back to their previous state and to allow the repeating of history in the event that data is lost and needs to be restored to a consistent state. Because of the importance of the transaction log to the consistency of the data on the system, it must be kept on some form of stable storage, generally one or more magnetic disks. To allow for recovery from media failure, the log is typically also archived to some type of backup device such as a magnetic tape. Thus the logging subsystem consists of two devices, the log device and the log backup device, as well as the interconnect algorithm between the two, which we term the log backup method.
If the logging subsystem is critical to an application, the bandwidth of the log can become a bottleneck to the throughput of the system. The purpose of this research is to break the log bottleneck at minimum cost. The bottleneck in the log may be at one or both of these devices or the interconnect algorithm between the two. To be able to break the log bottleneck, we must be able to increase the effective bandwidth of one or both devices while also selecting the best log backup method, at minimum cost.

In order to increase the effective bandwidth of the log device, we consider writing log records in parallel to multiple log devices (parallel logging [Agrawal85a, Kumar90]) as well as the use of striping [Salem86]. Our conclusion is that striping offers the best alternative, and we use Redundant Arrays of Inexpensive Disks (RAID) [Patterson88] as an extension to striping to provide maximum bandwidth at the least cost, while maintaining an acceptable level of reliability. Along with our discussion of RAID, we introduce a new RAID algorithm that is designed to overcome the performance problems of small writes in a RAID log.

In order to increase the effective bandwidth of the log backup device, we suggest the use of inexpensive tape drives and striping the log backup across the multiple drives. We recommend that the bandwidth of the log backup device be increased in this manner to the point that it reaches the effective bandwidth of the log device.

Although volumes of research have been done on how best to perform logging and recovery in a database system or file system [Agrawal85a, Agrawal85b, Agrawal85c, Doublis89, Franklin92, Gray81b, Haskin88, Jhingran92, Korth90, Kumar90, Mohan92, Reuter84, Rosenblum92, Seltzer90, Seltzer93, etc.], little has been published regarding what is the best interconnect algorithm between the log device and log backup device. Most papers dealing with the subject of logging and recovery
examine the many other important issues in this area without focusing on the issue of what to do when the log device fills. We present the novel approach of backing up the log synchronously, where the log backup device is essentially a mirror of the log device. We also examine the effects and costs of other log backup methods.

This thesis is organized as follows: In Chapter 2, the model for the various logging subsystem configurations is described along with previous research in this area. Our description of the logging model includes our criteria and weighting system for comparing two logging subsystem configurations. Chapter 2 also details the assumptions used in reaching our conclusions. Chapter 3 discusses the issue of log device bandwidth and presents our new RAID algorithm, while Chapter 4 discusses the issue of log backup device bandwidth. Chapter 5 deals with the issue of the interconnect algorithm between the log device and log backup device, including the argument for backing up the log synchronously as the method that provides the best performance at least cost. Finally, Chapter 6 presents the data from an implemented prototype of some logging configurations presented in the thesis.
Chapter 2

The Logging Model and Previous Work

This Chapter contains a discussion of the logging model used for this study. Figure 2.1 provides an overview of the model described in the following sections. It shows the various pieces of the logging subsystem, including the logging method, log device(s), log backup device(s), and log backup interconnect algorithm. We define a logging subsystem configuration as a specific instance of a logging subsystem, including a specified set of log devices and log backup devices and a particular interconnect algorithm. Log devices and log backup devices are defined in the following sections.

This Chapter describes each of the pieces of the logging model, including the logging method we will use, the hardware used for logging and log backup, and the log backup interconnect algorithms. Along with our discussion of each of the pieces of the logging model, we also describe previous solutions for increasing the effective bandwidth of the logging subsystem. This Chapter also presents our criteria for evaluating various logging subsystem configurations and describes the assumptions used in reaching our conclusions. We know of no other comparable models for describing and evaluating logging subsystems.
2.1 Logging Methods

The most widely used logging method today, and the one used here, is known as write ahead logging (WAL) [Gray78]. In write ahead logging, updates occur in place in volatile storage as a transaction executes. This is in contrast to techniques such as the use of shadow pages to contain updates until the commit of the transaction [Gray81b]. Along with the update to a page, one or more log records are written to reflect the changes to the page. The basic rule for the WAL protocol is that log records allowing for the undo (or rollback) of changes to data must be placed on stable storage prior to the modified data being placed on stable storage, replacing the previous version of the data. In order to satisfy the durability requirements at transaction commit, all log records pertaining to the redo (or roll forward) of the committing transaction must be on stable storage before the transaction is considered committed.
In the terminology given in [Haerder83], WAL is using the "STEAL" and "NO FORCE" data buffering policies for transaction execution and recovery. This means that modified buffers can be "stolen" from a transaction and written to disk prior to its commit, and transactions are "not forced" to write all modified data buffers to disk on commit. The logging and recovery implications of this method are that the log must include information to allow the undo of an update (e.g., a before image) to be used in the event of transaction rollback, and it must also include information to allow the redo of an update (e.g., an after image) in order to repeat history in the event that committed transactions have not had their modified data placed on stable storage at the time of a system crash (resulting in the loss of the contents of memory).

One current example of a WAL implementation is ARIES [Mohan92]. We will use the ARIES method of write ahead logging for our logging discussions because it is well known and used in practice (see [Haskin88, Franklin92, Jhingran92, Kumar90]) (Assumption 3). In the ARIES implementation of the WAL protocol, log records are given a log sequence number (LSN) and every page contains the LSN of the last update to the page. Before a page is written to non-volatile storage, all log records up to the LSN on the page must be written to stable storage first. Rollback is facilitated by linking the log records of a transaction in a backward chain, with each log record containing the LSN of the log record that precedes it in the transaction. Rollback is implemented by reading the log in reverse, following the backward chain, and undoing each undoable action as it is encountered. As a log record has its operation undone, a special Compensation Log Record (CLR) [Crus84] is written to indicate the undo. The CLR contains the previous LSN of the log

---

1 In the rest of this thesis, assumptions are referred to by the number associated with them in Section 2.5, which summarizes the assumptions.
record whose undo caused it to be written, thus it points to the next log record that will need to be undone. CLR’s are redo-only log records which are never rolled back.

During normal operation, ARIES maintains a transaction table that includes the state of the open transactions and the LSN of the last log record of each transaction. It also maintains a dirty pages table that lists the pages containing updates that have not yet been written to stable storage. Each entry in the table contains the LSN of the log record that first caused the page to be marked dirty, indicating the earliest log record that might need to be redone to repeat history during recovery. ARIES also takes periodic checkpoints [Gray81b] during normal operation and writes the transaction table and dirty pages table as part of the checkpoint log record.

ARIES uses a three pass system to evaluate the log and redo updates to bring the system back into a consistent state after a crash. The three phases of recovery are illustrated in Figure 2.2.

The first phase of recovery is known as the analysis phase. During this phase, ARIES determines the following:

- The log record to start with for the second (redo) phase.
- The pages that could have been dirty at the time of the crash.
- The transactions that were still open at the time of the crash.

Transactions that have not committed or reached the prepared state of a two phase commit [Mohan86] at the time of the crash will need to be rolled back at the end of recovery during the final (undo) phase.
The second phase is known as the redo phase. During the redo phase, log records are reapplied, even for committed transactions and transactions that will be rolled back at the end of recovery. A log record must be redone if its LSN is greater than the LSN on the page (or pages) that it affects.

After an update is redone, the LSN on the page is changed to the log record's LSN to indicate that this log record has been applied to the page. The redo phase also obtains locks needed to protect uncommitted updates of transactions that will be left in the prepared state [Mohan86] at the end of recovery. No log records are written during this phase of recovery.

The final phase of recovery is the undo phase. In this phase, all open transactions (not in the prepared state of a two phase commit) that are left over from the redo phase are rolled back, bringing the system into a consistent state. Log records are generated for this phase just the same as they would be during the rollback of transactions during normal operation.

Figure 2.2: Phases of ARIES Restart Recovery [Franklin92]
In the case of a hard crash of a data disk where all information on the disk is lost, the recovery will take place in a slightly different manner. First, the disk must have its data restored from a previous archive (assuming that the granularity of media recovery is at the disk level). Next, all log data since the archive that pertains to this disk is reapplied to bring it back into a consistent state. The archive will include the point in the log at which the roll forward should start. Since the starting point for rolling forward the log is well known, the analysis phase is not necessary and the recovery begins with the redo phase [Mohan92]. The redo phase is complete when all necessary log records up to the current point in time have been applied. The undo phase of recovery is then executed in the same manner as recovery from a soft crash.

2.2 Hardware

This Section describes the hardware which we consider as part of the logging model. It first describes the possible devices for containing log data and then describes the possible devices for backing up log data. After describing the hardware characteristics, each Section describes previous work in the area of increasing the bandwidth of the logging and log backup devices. To facilitate the comparison of various logging configurations, each type of device is assigned a maximum transfer speed and a cost in dollars. The values assigned for most of these characteristics were taken from several Transaction Processing Performance Council (TPC) benchmark full disclosure reports [HP-Oracle, NCR-Informix, Pyramid-Oracle, Sequent-Oracle, Sequent-Sybase] as well as the device characteristics given in [Drapeau93].
2.2.1 Log Devices

In the introduction, we indicated that the log must be kept on some form of stable storage, generally one or more magnetic disks. Stable storage is defined as either an inherently non-volatile medium such as a magnetic disk or magnetic tape, or a volatile medium that has its own self-contained power supply that allows it to transfer its data to a non-volatile medium after the failure of an external power supply [TpCB, Mohan92]. Transaction processing power is generally described as cost per transaction per unit of time [Anon85], so for reasons of cost the log is generally written to non-volatile storage. In order to facilitate the rollback of incomplete or failed transactions, random access is required in the log, which makes the logging medium of choice one or more magnetic disks rather than magnetic tape.

We will therefore consider a log device to be a particular grouping of magnetic disks that is treated by the logging subsystem as a single device. The four types of log devices described in this Section are:

- Single Disk
- Mirrored Disk
- Striped Disk
- RAID Disk

Any logging subsystem configuration is assumed to make use of only one of these types of log devices, i.e., it does not use both a single disk and a striped disk as log devices (Assumption 2). Our experience indicates that using multiple types of log devices is rarely done in practice, so for the sake of simplicity we only deal with configurations using a single type of log device. Due to durability requirements (see Section 2.1), a write to the log device is not considered to be complete until all the data written has been placed on the physical device (i.e., not simply copied into a disk controller cache or operating system buffer).
A single disk is very simply a single hard disk. For the sake of comparison, we will assume it has 1.5 Gigabytes (G) of storage capacity, a maximum transfer speed of 1.5 Megabytes (M) per second, and a cost of $10,000 (Assumption 4). These values were taken from the hardware we used in our prototype implementation in Chapter 6 as well as the average capacity and cost characteristics found in the various TPC benchmark full disclosure reports listed above.

A mirrored disk is a pair of single disks that are accessed in tandem [Bitton88]. Mirroring is assumed to be implemented in software by either the database system or the operating system at no added cost (Assumption 6). There are TPC full disclosure reports where mirroring is used in the log and implemented either by the database system [Sequent-Oracle, Sequent-Sybase] or operating system [Pyramid-Oracle] without any added cost to the benchmarked configuration. There is also evidence in product documentation [Informix6, SybaseSQL] to support this assumption.

A striped disk is made up of N single disks, where N > 1. It increases device bandwidth by striping the data across the N disks. Striping means that each block of data is divided into one or more sub-blocks (depending on block size) and the sub-blocks are written to one or more of the N disks in parallel [Salem86]. The sub-blocks could actually be some sort of interleaving of the data, as in [Kim86]. We will assume that striping is implemented in hardware at an added cost of $2000 per disk (Assumption 5). Striping has come to be known as RAID Level 0, and although RAID can be implemented in software [Patterson88], we will assume that it is implemented in hardware. This means that the user purchases a RAID device containing N disks as a "black box" that is connected to the system and appears as a single disk. This is in contrast to purchasing the disks and then buying or implementing a software package to turn the disks into a RAID. There are TPC benchmark
full disclosure reports using RAID for the log [HP-Oracle, NCR-Infor-
mix] where RAID is implemented in hardware at an added cost of about
$2000 per disk.

The other "levels" of RAID [Patterson88] take disk striping one step
further by adding parity information to eliminate a single point of
failure in the device. There were 5 "levels" of RAID in the original
RAID description [Patterson88], with Level 5 RAID being the most widely
used in practice (see [Lee91, Katz92, Katz93, Mourad92, Stone-
braker90]). In Level 5 RAID, data is interleaved at the block level,
where a block can be a page, sector, track, etc. One of the blocks in
the stripe across the N+1 disks contains the parity information for the
stripe. No single disk in the array will always be the parity disk,
with parity information being spread among each of the disks to avoid
bottlenecking on writes of the parity. Parity information is calculated
as the exclusive OR of all the corresponding data bits in the stripe.
We will assume that the same hardware that provides striping (RAID
Level 0) also provides RAID Level 5 (Assumption 5). Therefore, the
disks in a RAID device will be priced at $12,000 each, with the added
$2,000 per disk due to the hardware changes required to turn the inde-
pendent disks into a RAID (Assumption 5). The RAID device is capable
of transferring data at N*1.5 M/second and has a total storage capacity
of N*1.5G.

Although simple striping has come to be known as Level 0 RAID and the
mirrored disk described above is an instance of Level 1 RAID [Patter-
son88], in this thesis we will not distinguish among these RAID types
and only term a device a RAID if Level 5 RAID is being used.

We now discuss previous work in the area of log device bandwidth.
There have been several suggestions for increasing the effective band-
width of the log device. One is to perform what is known as group-
commit. This technique was first implemented in IMS FastPath [Gray91]
and was independently invented and described in [DeWitt84]. In group-commit, the write of the log to stable storage is delayed a short period of time until more log data is available to write to disk. The use of a disk array for the log is suggested in [Selinger90], along with a battery backed up memory to buffer log records until enough log data is available to perform a large write across the disk array, which is a variation on group-commit. Thus group-commit allows more of the bandwidth of the log disk to be utilized for each log write. Although this technique allows for maximizing the use of the bandwidth of the given log device, it overcomes the log device bottleneck in a limited way because it does not increase the bandwidth of the device itself.

A second technique for increasing the effective bandwidth of the log device is log compression. One method for log compression suggested in [DeWitt84] is to not write log records required for rolling back incomplete or failed transactions to disk, and only write the log records required for the repeating of history for restoring a system to a consistent state after the loss of a data disk. This increases the effective bandwidth of the log device by requiring that less information be written to the log. Their method assumes that a battery backed up memory is available to hold log records prior to commit, allowing for rollback in the event of a crash that results in the loss of the contents of memory. This technique is implemented in Oracle RDBMS™[Oracle6a], but they do not rely on battery backed up memory. Instead, they have a separate log area to contain rollback information. The rollback information for a transaction is only flushed to disk if a page that has been modified by the transaction must be flushed to disk prior to the commit of the transaction (see Section 2.1). A different type of log compression is mentioned in [Mohan92]. This method is to log only the differences between the before and after images of a changing piece of data rather than logging complete before and after images to allow for the
undo (rollback) or redo of the transaction. Log compression is similar to group-commit in that it allows for maximizing the bandwidth of the log device, but it also does not prevent the log device from becoming a bottleneck as it does not increase the bandwidth of the device itself.

A third method for increasing log device bandwidth is to perform what is known as parallel logging. This method is recommended by [Agrawal85a, Agrawal85b, Kumar90]. In parallel logging, multiple independent log devices are filled simultaneously. As log records are generated, they are distributed across the independent log devices according to some algorithm such as round-robin (which behaves similar to striping) or based on a transaction identifier. In this method, log records are buffered and written independently to the multiple log disks, which is in contrast to striping, where all log records are placed in the same buffer and written with a single write across one or more of the striped disks. This technique is also described in some form in [DeWitt84], although it is not termed parallel logging. Parallel logging is also suggested in [Gray91] as a way to increase the effective bandwidth of the log. Note, however, that they also suggest parallel logging for overcoming other potential bottlenecks in the log such as the network or the log latch.

Although parallel logging increases the bandwidth of the log device, our conclusion is that the use of a striped log device is the best option for overcoming the bottleneck of log device bandwidth. We'll postpone our analysis of the difficulties inherent in parallel logging and how striping overcomes these difficulties to Chapter 3. We do not discuss group-commit or log compression further because neither method increases the bandwidth of the log device.
2.2.2 Log Backup Devices

We indicated above that the log is typically archived to some type of backup device, such as a magnetic tape. Backing up the log is not strictly required, and the implications of not backing up the log are dealt with in the following discussion. The backup device does not have to be a tape drive, but they are the most common and are the only ones dealt with in this thesis. We will make use of streaming tape drives for log backup because they are well suited to the purpose. Similar to log devices, we define a log backup device as a particular grouping of magnetic tape drives that is treated by the logging subsystem as a single device. We will consider five different types of log backup devices. These are:

- No Log Backup
- Single Tape
- Mirrored Tape
- Striped Tape
- RAID Tape

First we consider using no log backup device. Observe that we must maintain log data between archives of the database, which implies that a log device cannot be reused for new log data until its log data has been backed up or the database has been archived. The reason for keeping the log data is that recovery from the loss of a data disk requires restoring the data from a previous archive followed by reapplying updates to the data since the archive, using the information contained in the log (see Section 2.1). It is unreasonable to assume that an archive of the database will occur more often than once per day, so with no log backup device, we must have enough log devices to contain a full 24 hours of log data. Now consider the following example. At a low average logging rate of 250K/second, we would be required to maintain over 21.5G of log data on disk, requiring 15 single log disks.
Maintaining this much log data on disk makes not backing up the log very cost prohibitive even without providing redundancy such as mirroring in the log devices. Because of this, we consider the option of having no log backup device to be infeasible in our model and ignore it throughout the remainder of this thesis.

A single tape device is simply a single streaming tape drive. We will assume that the tape drive has 5G of data capacity with a maximum transfer speed of .5M per second and a cost of $3,000 (Assumption 7). These numbers were taken from [Drapeau93] and are the same as the log backup device used in our prototype implementation in Chapter 6. It is also assumed that the tape drive has a cache of 1M, and that it only flushes data to the tape when the cache is full, a write is requested of a special block such as an End Of File marker, or due to the closing of the tape drive (Assumption 8). This is typical of most streaming tape drives, such as [Exa8500, Megatape]. Because of this cache, the completion indication from a write operation to the log backup device may simply indicate the success of copying the data into its cache, and is not necessarily the result of actually writing it to the physical tape.

One more issue that must be dealt with when discussing streaming tape drives is the amount of time it takes to rewind, eject, and load tapes, as described in [Drapeau93, Exa8500, Megatape]. Many tape drives require the rewinding of the current tape prior to ejecting the tape, and a streaming tape drive can require from 2 to 3 minutes to rewind the tape. Further, it can take from 15 to 45 seconds just to eject the tape from the drive. Simply loading a tape can also take from 15 to 35 seconds. For drives that require rewinding prior to ejecting the tape, this adds up to requiring between 2.5 and 4.5 minutes to simply change tapes in the drive. This backup device "down time" must be taken into account in discussing log backup.
A mirrored tape is simply a pair of single tapes that are accessed in tandem. Configuring mirrored tape drives to appear as a single device is assumed to be implemented in software through the purchase of a software package (Assumption 9). We assume a software implementation because tape mirroring is typically implemented in software. An example of a software product which does this is Pyramid's FastTRAC™ [Pyramid]. It allows multiple tape drives to be configured as a single device that appears as either a RAID Level 0 (simple striping) or a RAID Level 4 (one device used for parity). This special software package will be assumed to cost $10,000² (Assumption 9). RAID Level 4 using two tape drives implements mirroring since the parity drive acts as a mirror of the other drive. The hardware cost for a mirrored tape device is 2*$3,000, with the same maximum transfer speed and capacity as a single tape device.

A striped tape device is N single tape drives that have been configured such that all N drives can be read from or written to in parallel. Striping will be assumed to be implemented using the same software package that implements mirroring (Assumption 9). The hardware cost of the striped tape device is N*$3,000, with a maximum transfer speed of N*.5M per second and a maximum capacity of N*5G.

The RAID tape device will also use this same software package (Assumption 9), and will add one extra drive for parity and use RAID Level 4. In RAID Level 4, data is interleaved at the block level but one drive always serves as the parity drive. The hardware cost for using RAID Level 4 will be N+1*$3,000, with the same maximum transfer speed and capacity as a tape device using simple striping.

² Although the price of $10,000 is a reasonable cost approximation for a software package of this type, it is intended for comparison purposes only and is not intended to reflect Pyramid's pricing structure for FastTRAC.
The reliability of magnetic tapes could be a problem for recovery in a RAID tape array. If multiple tapes in the array fail, there will be a loss of backup data to the system. Although it is possible that failure rates for tapes are high enough to warrant concern when using a RAID tape array [Gray94], our own industry experience indicates that tapes are not so unreliable as to prevent RAID tapes from being a viable option for log backup.

One more item that should be mentioned concerning striped tapes is the issue of when to switch the tapes in the drives. The authors in [Drapeau93] point out that bad spots on tapes cause tape drives to retry operations on subsequent areas of the tape until they are successful. This can cause the current write position for the tapes in a striped tape array to be out of sync. It is unclear how much of a problem this is in practice. For the sake of simplicity, we don't take this possible write performance delay into account in our comparisons and assume that we will change all of the tapes in the array as soon as one of them reaches the end of the tape (Assumption 10).

In terms of previous work on the impact of log backup device bandwidth on the performance of the logging subsystem, there appears to be nothing in the literature that deals specifically with this issue. The authors in [Gray91] recognize that the log must be backed up, but they don't deal with the bandwidth of the log backup device. Perhaps the impact of a log backup device that is slower than the log device is obvious. Nevertheless, we deal with this issue and examine how a slower log backup device affects the performance of the logging subsystem for each of the log backup interconnect algorithms. Our recommendation for increasing log backup device bandwidth will be to use striped tapes such that the log backup device has the same effective bandwidth as the log device. We discuss this issue and give support for our conclusions in Chapter 4.
2.3 Log Backup Methods

This Section describes the log backup methods being studied. A log backup method is the interconnect algorithm between the log device and the log backup device and can be defined as the algorithm for transferring the log data from the log device to the backup device. After describing the backup methods we will examine, we present what can be found in the literature on this topic and what current systems use each interconnect algorithm based on what is found (or implied) in the literature and in product documentation. We will examine four methods for dealing with log backup. These are:

- Asynchronous Backup
- Synchronous Backup
- Offline Backup
- Staggered Allocation Backup

The asynchronous log backup method backs up log data from the log device to the log backup device while the logging subsystem is concurrently streaming log records to it as the current log device (Figure 2.3). This means that these two activities are competing for the use of the log device. This competition causes extra seeks on the disk(s) as the disk arm(s) will move back and forth between write requests of new log data and read requests for the backup of previously written log data. The seek distance will depend on the difference in bandwidth between the log device and the log backup device. If the log backup device is fast enough to keep up with the log device, the seek distance will be kept at a minimum. If the log backup device is slower than the log device, the asynchronous log backup will fall further behind the current logging position, causing much longer seeks in the log device than normal and a greater detrimental impact on logging performance. Elimination of seeks in a log device was claimed to give better performance by as much as an order of magnitude in [Douglast89]. However,
this claim was based on very small random write operations where seek time will completely overwhelm transfer time. In our experience with larger transfers, we have not seen degradation that is nearly so severe. Our results (see Chapter 6) indicate that the slowdown in maximum average and burst speeds when the log backup device is able to keep up with the log device is about 10 to 15%, and when the backup device is significantly slower than the log device the slowdown in maximum burst speed averages about 40%. We've also found that when the backup device is slower than the log device the slowdown in maximum average speed is around 5% compared to backup device speed. The discussion of our implemented prototype in Chapter 6 gives further details as to how we arrived at these numbers.

![Figure 2.3: Asynchronous Log Backup Method](image)

The second log backup method is termed synchronous backup. In this method, the log data is written to both the log device and log backup device in parallel from the log buffer (Figure 2.4). Note that due to the cache utilized by the backup device (see Section 2.2.2), the data
sent to the backup device is not necessarily on stable storage at the point that a successful status is returned for the write operation. However, for the sake of simplicity, we will assume that we wait for a status from each write operation to the backup device (Assumption 11). Because the log is being written synchronously to the log backup device, changing tapes in the backup device will cause the logging subsystem to come to a complete halt until the change is complete and the backup can continue. To prevent this, the synchronous interconnect algorithm requires the availability of a second log backup device. When it is time to change tapes, the second backup device becomes the current log backup device. While the tapes in the new current device are being filled with log data, the tapes in the other device are changed and prepared for the next switch of log backup devices.

Figure 2.4: Synchronous Log Backup Method

3 The second backup device is not always active, but we do not make any assumptions about it being used for other applications.
The third log backup method is termed offline backup. In this method, we back up a log device after moving to the next instance of the log device (Figure 2.5). After an instance of the log device is filled completely, the next instance of the log device becomes the current one. After a log device is no longer current, the log backup is allowed to archive the log data from the device. Note that this method is dual to synchronous backup in that it requires the purchase of two instances of the log device as opposed to requiring the purchase of two instances of the log backup device. Also note that unless redundancy is being used in the log device (i.e., mirroring or RAID), a failure in either the current log device or the previous log device prior to it being completely backed up will result in a loss of data to the system.

Figure 2.5: Offline Log Backup Method

The final log backup method is a variation on offline backup presented in [Gray91] and shown in Figure 2.6. It is an improvement on the above described offline backup method when mirroring is to be used in the log device. In this method, called staggered allocation, three log
files are allocated across three disks, with the ith log file allocated to disks \( D_j \) and \( D_k \), where \( j = i \mod 3 \) and \( k = (i + 1) \mod 3 \). The current log file is mirrored on two of the disks, with the most recently completed log file present on the free disk. This most recently completed log file is backed up while the current log file is written to the other two disks. Thus offline backup is implemented using mirroring with only three disks instead of the normal four (two mirrored pairs), reducing the cost of the logging subsystem. We will use this improved offline method when examining configurations using offline backup and mirrored log devices.

![Figure 2.6: Staggered Allocation Log Backup Method [Gray91]](image)

Next we discuss previous work with respect to the log backup interconnect. Similar to log backup device bandwidth, the issue of the log backup interconnect algorithm is also not dealt with in detail in the literature. The authors in [Gray91] recognize the performance problems caused by asynchronous backup, but they don’t attempt to quantify this impact. Similarly, the documentation for INFORMIX-Online Dynamic Serv-
er™ [Informix6] also warns of the performance impact of contention for the log device and suggests offline backup be used. Commercial or prototype systems using asynchronous backup include IBM's DB2™ [Crus84], IBM's QuickSilver [Haskin88], INFORMIX-OnLine Dynamic Server™ [Informix6], Oracle RDBMS™ [Oracle6b], Sybase SQL Server™ [SybaseSQL], and System R [Gray81b]. This method is also implied in the literature for [Douglis89] and [Seltzer90]. Offline backup is also used by DB2, Informix, and Oracle, according to the above references.

4 No current systems that we know of use the synchronous backup method.

Our conclusion for the optimal interconnect algorithm is to use the synchronous backup method. We conclude that as long as the cost of a log backup device is less than the cost of a log device, the synchronous interconnect algorithm provides the best performance at least cost. Our analysis of the log backup interconnect is contained in Chapter 5.

2.4 Evaluation Criteria

In order to evaluate and compare logging subsystem configurations including hardware and interconnect algorithm, we propose the use of eight evaluation criteria. For each criterion, we classify a configuration as either Excellent, Average, or Poor. We also provide a system for weighting the criteria based on our opinion of its importance. The weights help to determine the better of two configurations. The criteria are:

1. Loss of data for any single failure.
2. Maximum average speed.

4 DB2, Informix, and Oracle all allow the user to allocate log files in any size and number and place them as desired on disk. They also allow the backup of any log file after it fills. Thus they allow either asynchronous or offline backup depending on whether multiple log files are placed on a single disk or each log file is placed on its own disk.
4. Cost per performance for the configuration.
5. Recovery time for a soft crash.
6. Recovery time for a hard crash.
7. Special hardware changes required to support the configuration.
8. Special software changes required to support the configuration.

These criteria are considered to be sufficient to compare two logging subsystem configurations to determine which one is better based on our criteria weighting system. Each criterion is explained below, including how we classify a configuration as Excellent, Average, or Poor for each criterion. After describing the criteria, we explain how we weight the criteria to compare configurations. Table 2.1 summarizes the information detailed in the following sections.
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Excellent</th>
<th>Average</th>
<th>Poor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum average speed.</td>
<td>Equal to log device bandwidth.</td>
<td>&gt; 50% of log device</td>
<td>&lt;= 50% of log device</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>Equal to log device bandwidth.</td>
<td>&gt; 75% of log device</td>
<td>&lt;= 75% of log device</td>
</tr>
<tr>
<td>Recovery time for hard crash.</td>
<td>No recovery needed for log system crash, read = log device for data disk crash.</td>
<td>Read = log device, no archive required for tape drive crash.</td>
<td>Read &lt; log device, loss of data, or archive required.</td>
</tr>
<tr>
<td>Special hardware changes.</td>
<td>None required.</td>
<td>Hardware available.</td>
<td>Hardware unavailable.</td>
</tr>
<tr>
<td>Special software changes.</td>
<td>None required.</td>
<td>Software available.</td>
<td>Software unavailable.</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of Evaluation Criteria

### 2.4.1 Loss of Data

The loss of data for any single failure is separated into its own category due to its importance. We require the database system to be able to survive any single failure without a loss of data, including both soft and hard crashes. This is also required by the industry standard TPC benchmarks \([\text{TpcA, TpcB, TpcC, TpcD}]\).

We define a soft crash as an interruption in the operation of the system that results in the loss of the contents of memory \([\text{Mohan92, TpcA, TpcB, TpcC, TpcD}]\). In our logging model, no configuration will
lose data in the event of a soft crash. This is due to the use of the write ahead logging (WAL) protocol [Gray78] as the logging method. WAL guarantees that log records required for undo are on stable storage prior to the modified data they refer to and log records required for redo are all on stable storage at commit.

We define a hard crash as the loss of any one durable medium [TpcA, TpcB, TpcC, TpcD], including any single data disk, log disk, backup tape drive, or backup tape. It is possible for some configurations in our model to lose data due to a hard crash. For example, if we are using a single disk for the log device and backing up the log asynchronously, a hard crash in the log disk will result in a loss of data to the system. The amount of data lost will be the difference between the current log write position for new log data and the current log read position for backing up old log data.

We will classify a configuration as Excellent for this criterion if there is no loss of data for any single failure. We will classify a configuration as Poor if there is a loss of data on the system for any single failure. Since there is no middle ground between losing and not losing data, we don’t classify any configuration as Average for this criterion.

2.4.2 Maximum Average Speed

The speed measurements are intended to determine the transaction processing bounds placed upon the database system by the logging subsystem.

The maximum average speed indicates the maximum sustainable logging rate for a given logging subsystem configuration. We will make the assumption that the database system is required to run 24 hours a day, 7 days a week (i.e., there is very little slow or down time) (Assumption 1). This is becoming a frequent requirement in the industry. We will
classify the maximum average speed for a configuration as Excellent if it is the same as the maximum bandwidth of the log device. We will classify this performance as Average if it is greater than 50% of the maximum bandwidth of the log device. A configuration is considered Poor for this criterion if its average speed is less than or equal to 50% of the maximum bandwidth of the log device.

For example, consider a configuration using synchronous log backup and a single disk for the log in order to sustain a 1.5M/second log rate. When using synchronous backup, the maximum average speed will be the slower of either the log device or log backup device. If we are using a single tape drive for the log backup device, our maximum average speed will be the speed of the tape drive, or 0.5M/second. This maximum average speed will be rated as Poor since it is only one-third of the required speed of the log device. If we use striping and add a second tape drive, we achieve a maximum average speed of 1M/second and are classified as Average for the criterion. Adding a third tape drive allows the configuration to maintain a maximum average speed of 1.5M/second, making it Excellent for this criterion.

2.4.3 Maximum Burst Speed

The maximum burst speed indicates the maximum speed that can be attained for a (possibly) short period of time by the logging subsystem. This will be greater than or equal to the maximum average speed, and is an indication of what a logging subsystem configuration is capable of during times of peak activity on the system. A configuration will be considered Excellent for this criterion if the maximum burst speed is equal to the maximum bandwidth of the log device. It will be considered Average if it performs at greater than 75% of the maximum bandwidth of the log device and Poor if it performs at less than or equal to 75% of the maximum bandwidth of the log device.
2.4.4 Cost/Performance

The cost to support the configuration is determined by the total hardware and software cost for the configuration, including special hardware and software. For the sake of simplicity, we exclude peripheral hardware such as cables, controllers, power supplies, and magnetic tapes, as well as other costs such as maintenance and support. When comparing the cost per performance of two logging configurations, they are compared based on both the cost per maximum average speed and the cost per maximum burst speed. This is similar to benchmarking the two configurations and determining which configuration allows the best transactions per second (TPS) per dollar. It is analogous to measuring the throughput per dollar for the system.

Because cost is a relative measure, it is useful only when comparing two configurations to determine the better of the two. Therefore, we do not evaluate the goodness of the cost/performance of a single configuration and do not classify cost/performance as Excellent, Average, or Poor.

2.4.5 Soft Crash Recovery

Recovery time for a soft crash is intended to measure the amount of time required to return the system to a consistent state following a soft crash. Recall from Section 2.1 that recovery from a soft crash in the ARIES [Mohan92] logging model involves three phases:

1. The analysis phase
2. The redo phase
3. The undo phase

The analysis phase determines which pages could have been dirty at the time of the crash as well as the log record with which to begin the redo phase. The redo phase is where all log records are reapplied, even
for committed transactions and those that will be rolled back at the end of recovery. The undo phase will undo all open transactions left over after the redo phase.

The authors in [Jhingran92] studied recovery time in an ARIES-based WAL environment in some detail, including the effect of log read time on total recovery time. They concluded that recovery time is dominated by data I/O during the redo phase of recovery. Log read time is bounded by the oldest dirty page in the buffer pool (i.e., the oldest update not yet flushed to disk). The oldest update not yet flushed to disk bounds the log read time because it is the point at which the redo phase of recovery will begin. Log read time was found to be much less significant than I/O time for data by nearly an order of magnitude. The authors noted, however, that as buffer sizes get bigger, dirty pages will tend to be kept in memory longer and make log reading more of a factor in recovery.

For the sake of simplicity, we will only consider log read time when comparing two configurations for this criterion. We will assume that both the processing time for the log data and the amount of log data to read is the same for each configuration and simply compare the speed at which we can read the log device (Assumption 12).

Note that each of the three interconnect algorithms must deal with the loss of the log backup device's buffer of log data in the event of a soft crash such as a power failure causing the loss of the contents of this memory. Because the backup device is buffering data, the system must periodically record in stable storage the log position of the last log data known to be on the physical backup device. This is the point from which log data will be backed up following a restart due to a soft crash. Some amount of log data will have to be backed up again as part of recovery, regardless of the interconnect algorithm. The synchronous interconnect algorithm has a special requirement in that it must com-
plete this catching up of the log backup before the system becomes fully operational and is able to backup log data synchronously again. However, this rewriting of log data can occur in conjunction with the redo phase of recovery as the log is read, so the increased impact on recovery compared to the other methods should be minimal. Because of this, we do not take the interconnect algorithm into consideration for this type of crash and simply compare log read time.

We will classify the soft crash performance as Excellent if multiple log disks can be read in parallel during crash recovery. We will classify it as Average if we are simply able to read the data at the maximum bandwidth of the log device, and Poor if for some reason we are unable to use the full bandwidth of the log device to read log data. In our particular model, we will only be dealing with configurations that are either Excellent or Average for this criterion.

2.4.6 Hard Crash Recovery

Recovery time for a hard crash is intended to measure the amount of time required to return the system to a consistent state following a more serious hard crash. The recovery strategy and our method of evaluation for this criterion will vary depending on the type of hard crash.

For a hard crash in a data disk, we will need to restore the data from an archive and then process all log data since the beginning of the archive to reapply the changes to the data, restoring it to the current state (see Section 2.1). Although some of the log data required will still be on disk, much of the log data will be on log backup tapes, so we will be reading and processing log data directly from a tape device. If we assume that an archive occurs no more often than once per day, even a slow log rate of 250K/second would require a little more than 4 log backup tapes over a 24 hour period (21.6G). Assuming that a crash of a data disk occurs half way between archives on average, we
still require about 10G of log data to be read in and processed, which is several times the size of a single disk. For this reason, when comparing two logging subsystem configurations for this type of crash, we will only compare the rate at which we can read log data from the backup tapes. Similar to soft crash recovery, we are ignoring the amount of time required to restore the data to the failed disk. We are also ignoring log read time for log data still on disk. We compare log read performance from disk for our soft crash criterion.

For a hard crash of a single disk in a log device that does not have any redundancy such as mirroring or RAID, we need to restore the log data from the backup tape directly to the replaced log disk. If there is no redundancy in the log device and the data has not yet been backed up at the time of the crash, there will be a loss of data to the system.

For a hard crash in a tape drive, if there is redundancy such as mirroring or RAID in the log backup device, then the log backup continues uninterrupted while the failed drive is being replaced. If striping is being used in the log backup device, then the log backup can continue while striping the backup across fewer tape drives. This assumes that the log data on the tape in the failed drive is still intact and does not need to be backed up again due to the tape drive crash (single failure). The backup will be slower in this case and cause a slowdown in the maximum average speed of the system, but the backup can continue. Note that we will have to re-backup a small amount of log data in this case due to the loss of the failed tape drive’s buffer of log data. If we are using a single tape drive for the log backup device and it fails, the log backup stops until the drive can be replaced. After it is replaced, the log data written since the crash must be backed up. The alternative is to take an archive of the database after the log backup device is replaced and begin backing up the log data from that point forward.
A hard crash in a magnetic tape such as a tape breaking or a bad tape have similar implications for the database system. If there is redundancy such as mirroring or RAID in the log backup device, then the log backup continues uninterrupted while the failed media is being replaced. Otherwise, log data from the failed tape must be backed up again. If some of the log data on the failed tape has already been overwritten on disk, then we will be unable to back up the overwritten log data. At that point, in order to ensure recoverability from a hard crash in a data disk, we must archive the database and restart the log backup from the point of the archive. An archive is likely to be required due to the fact that a log backup tape has a capacity that is over three times that of a single disk (5G vs. 1.5G), making it likely that a disk that has been backed up to the tape has been reused for new log data at the time of the failure of the tape.

We will classify a configuration for this criterion based on what strategy is required to recover from each hard crash, which is a way of measuring the impact of the various crashes on the operation of the system. We will classify hard crash performance as Excellent if there is redundancy provided such that no recovery time is required for any hard crash in the logging subsystem, and we can read back the log at a rate equal to the maximum bandwidth of a log device for a hard crash of a data disk. We will classify it as Average if we are able to read the log from tape at a rate equal to the speed of a log device for both a data disk crash and a log disk crash, and no archive is ever required for recovery from the crash of a tape drive. We will classify this criterion as Poor if a loss of data can occur due to any hard crash, we are unable to read back the log at a rate equal to the maximum bandwidth of a log device, or if an archive can be required for the crash of a tape drive.
Table 2.2 summarizes the strategies for dealing with the different types of hard crashes when no redundancy is provided for the failed media.

<table>
<thead>
<tr>
<th>Type of Hard Crash</th>
<th>Strategy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Disk Crash</td>
<td>Read log from tape.</td>
</tr>
<tr>
<td>Log Disk Crash</td>
<td>Read log from tape.</td>
</tr>
<tr>
<td>Tape Drive Crash:</td>
<td>1) Re-backup lost buffer and log data since crash OR 2) Archive database</td>
</tr>
<tr>
<td>Single Tape Backup</td>
<td></td>
</tr>
<tr>
<td>Tape Drive Crash:</td>
<td>Re-backup lost buffer and continue.</td>
</tr>
<tr>
<td>Striped Tape Backup</td>
<td></td>
</tr>
<tr>
<td>Magnetic Tape Crash</td>
<td>1) Re-backup log data from lost tape OR 2) Archive database</td>
</tr>
</tbody>
</table>

Table 2.2: Strategies for Hard Crash Recovery Without Redundancy

2.4.7 Special Hardware

Because database administrators are concerned about new systems to support, we list special hardware changes separately to give an indication of any special hardware features used in the particular configuration. The criterion for special hardware is classified as Excellent for a configuration if it requires no special hardware. It is classified as Average if special hardware is needed but is available to be purchased. The configuration is classified as Poor for this criterion if it requires some special hardware, but the special hardware cannot be readily purchased and must be built by the developer of the system. None of the configurations that we are studying will be classified as Poor for this criterion.
2.4.8 Special Software

We list special software separately for the same reason we list special hardware separately. Similarly, special software will be classified as Excellent if no special software is required, Average if special software is needed but can be readily purchased, and Poor if the special software required must be written by the developer of the system. Again, none of the configurations that we are studying are classified as Poor for this criterion.

2.4.9 Weighting the Criteria

In order to facilitate the comparison of two logging configurations to determine which one is better, we will make use of a weighting system for the criteria. Each criterion is weighted according to our opinion of its importance. As we compare two configurations, the winner of the two for a particular criterion receives the points for the criterion. If the two are equal, then both receive the points. There are a total of 40 points in our sample weighting system, and the weights are given in Table 2.3. The reader may have his or her own priority and weights for these criteria, and may come up with their own score when comparing two configurations. When performing a comparison, we provide the total
score as well as the ratio of the score of the winner in the comparison versus the score of the loser, to provide an indication of how much better the winner was in the comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>40</strong></td>
</tr>
</tbody>
</table>

*Table 2.3: Log Criteria Weights*

Based on our assumption that the system is intended to have very little slow or down time (Assumption 1), we weight the maximum average speed to be as important as maximum burst speed.

Note that we give lost data the most weight. We consider data lost as much more expensive than any of the speed or cost characteristics of the configuration. By making the weight for lost data equal to all the other weights combined, we guarantee that a configuration that loses data will never be called the winner in a comparison with a configuration which does not lose data.

### 2.5 Assumptions

This Section is a summary of all the assumptions used in the evaluation of the various combinations of devices and interconnect algorithms described above. The list of assumptions is organized based on the justification for the assumption.
• Assumptions based on industry experience:
  • ASSUMPTION 1:
    The database system is required to operate 24 hours a day, 7 days a week.
  • ASSUMPTION 2:
    Any logging subsystem configuration uses one type of logging device.
  • ASSUMPTION 3:
    All logging configurations are using an ARIES [Mohan92] WAL protocol as the logging method.
• Assumptions based on TPC reports and product documentation:
  • ASSUMPTION 4:
    Any single log disk has 1.5G of storage capacity, a speed of 1.5M/second, and a cost of $10,000.
  • ASSUMPTION 5:
    Disk striping or RAID is implemented in hardware and costs an extra $2,000 per disk due to the hardware changes required to turn the disks into a striped (RAID Level 0) device or a RAID Level 5.
  • ASSUMPTION 6:
    Disk mirroring is implemented by the database system or operating system and comes at no extra cost.
  • ASSUMPTION 7:
    Any single tape drive has 5G of storage capacity, a speed of .5M per second, and a cost of $3,000.
  • ASSUMPTION 8:
    Each tape drive contains a 1M buffer used to buffer I/O to the device.
• ASSUMPTION 9:
  Tape mirroring, striping, or RAID Level 4 is implemented via a software package and costs an extra $10,000.

• Assumptions used for the sake of simplicity:

• ASSUMPTION 10:
  All tapes in a tape device are changed as soon as one of the tapes fills.

• ASSUMPTION 11:
  When performing synchronous backup of the log device, we will wait for a status from both the write to the log device and the log backup device before considering the write complete.

• ASSUMPTION 12:
  When comparing logging configurations, the time spent processing log data during recovery is assumed to be equal for each type of crash.
Chapter 3

Log Device Bandwidth

This Chapter discusses the issue of log device bandwidth (see Figure 2.1). We compare a logging subsystem configuration that increases log device bandwidth using parallel logging with a configuration using striping to increase log device bandwidth. Because we are principally concerned with the bandwidth of the log device in this Chapter, we do not make use of our criteria and weighting system in our comparison. Instead, we discuss the drawbacks inherent in parallel logging and then describe how striping is able to overcome most of these shortcomings. We then discuss the use of RAID in the log as an extension to striping for added reliability. Along with our discussion of RAID as a log device, we present our Append-Only RAID algorithm for optimal use of a RAID log.

3.1 The Parallel Logging Solution

The most obvious solution for increasing the bandwidth in the log device is to write to multiple independent log devices in parallel. This was described briefly in Section 2.2.1. In [Agrawal85a] and [Agrawal85b], the authors studied the issue of writing to multiple log devices in parallel. They concluded that when logging physical data (as opposed to logical operations such as predicates or update statements [Mohan92]), the log must be parallelized to prevent it from becoming a bottleneck. They proceed to examine various methods for distributing log records across log devices. Kumar [Kumar90] also studied the issue
of log device bandwidth and concluded that parallel logging is necessary to prevent the tail of the log from being a "hot spot", since it is continually being appended to.

To our knowledge, no current database system performs this type of parallel logging, even though no database system we know of performs purely logical logging. We believe that this is due to some drawbacks inherent in using multiple log disks in parallel.

The first drawback is in the area of reliability. When using N independent log disks, a hard crash in any one of the N disks results in a loss of data to the system. This means that the Mean Time To Failure (MTTF) of the logging subsystem due to disk failure is (MTTF of a single disk) / N, which is significantly less than the reliability of a single disk.

The solution to the reliability problem is to provide some type of redundancy at each log disk. However, this leads to a second drawback in the area of cost. Providing redundancy at each disk means providing at least a mirror for each log disk, resulting in a doubling of the cost for log devices. The overhead is essentially 100% for the needed redundancy.

The third drawback is the added implementation complexity of parallel logging, both at transaction commit and crash recovery. This was recognized in [Gray91] as part of the recommendation to use parallel logging. The authors in [Agrawal85a] show that distributing the log records across log disks based on the transaction identifier (i.e., sending all of a transaction's log records to the same disk) is not the best algorithm for distributing log records. Not sending all log records for a transaction to the same disk implies that at transaction commit the logging subsystem must ensure that all disks that had log
records sent to them for the committing transaction flush their log records prior to certifying the transaction as committed, adding extra overhead and complexity to the commit process.

Further difficulties arise at recovery time when using parallel logging. Recall that part of crash recovery is the repeating of history by reapplying log records to bring the system back into a consistent state. Reapplying log records is made more difficult due to the fact that unless log records are distributed based on the location of the data they refer to (i.e., sending all log records for disk X to log Y), the log records cannot be read from disk in the order in which they were originally applied to the data. This means that recovery must be augmented by a merge of the log records to get them into the proper order for repeating history. A merge of log records based on LSN is not difficult to implement, but it does add extra overhead and complexity to the recovery process. The authors in [Agrawal85b] claim that their parallel logging scheme does not require this merge of log records at recovery. However, this claim appears to be based on the assumption that locks can occur at no finer granularity than the page level, which is too restrictive for many database applications (see [NCR-Informix] for a TPC benchmark example). Kumar [Kumar90] allows for the necessity of a log record merge for recovery, but believes this added overhead and complexity is offset by being able to read from multiple log disks in parallel during recovery.

A final drawback to parallel logging is the difficulty in backing up the log disks. Performing offline backup at each disk is very cost prohibitive since it requires the addition of another N log disks. For staggered allocation backup, a total of three disks must be provided at each parallel log. In either case, the cost of the log disks is increased for offline backup. Asynchronous log backup at each log disk is a possibility, but it defeats the purpose of parallel logging in the
first place due to the performance problems caused by the contention for the log device in asynchronous backup. Synchronous backup at each log disk removes the need for mirroring the log disk, but requires two log backup devices at each log disk. Whatever method is used to back up the log, the backup of multiple independent log disks, each of which is being filled at a different rate, is difficult to manage.

One more item of note is that the log backup options presented don’t solve the problem of log record merging during the restore after a hard crash in a data disk. Recall that this type of restore requires the reapplying of all log data since the last archive of the media being restored. If log records are not distributed based on the location of the data they refer to, then unless there is a merge of the log records as they are placed on the backup tapes we must incur the merge overhead at recovery time, the same as recovering from disk following a soft crash.

We believe that these reliability, cost, implementation, and log backup difficulties have caused developers to avoid using parallel logging in the database system. The next Section presents the solutions involving striping which overcome these implementation problems.

### 3.2 The Striping Solution

We propose that the best way to increase the bandwidth of the log device is by striping the log across multiple disks. Simple striping is able to prevent the tail of the log from becoming a "hot spot" by striping the data across N log disks rather than always appending the log to a single disk. Kumar [Kumar90] claims that striping is unable to overcome this problem, but his claim is based on a log write size of one (2K) page, which is an unnecessary restriction (see [NCR-Informix]). On the contrary, performing one write across N log disks should always increase the bandwidth of the log device the same as N indepen-
dent writes to the N log disks. Further, one could argue that a balanced
distribution of log records in parallel logging that fills each disk
at about the same rate effectively reduces parallel logging to striping
in terms of device bandwidth. This is because in a heavily loaded sys­
tem, transaction commits will constantly request the flush of their log
records to disk, typically causing each independent log disk to have
its data flushed in parallel to the other log disks.

Striping is also able to overcome the implementation complexities in­
herent in parallel logging. Since all log records are being sent to
one logical device, there is no added complexity at transaction commit.
There is also no need for a merge of log records at recovery time, since
the records are read in from the striped device in the same order as
they were written. The read of log records can also proceed in parallel
as each read utilizes all of the log disks, as was the case for parallel
logging.

The fact that using striping reduces the logging subsystem to writing
to a single log device with N disks that are all being filled at the
same rate greatly reduces the complexity of managing the backup of the
log device as compared to parallel logging. We discuss which backup
method is best in Chapter 5.

Unfortunately, simple striping does not overcome the problems of
reliability and cost. The reliability in striping is no better than was
described above for parallel logging. If any one of the N log disks
fails, there is a loss of log data. If we are using synchronous backup,
this loss of data can be avoided, but this method requires a restore
of the log data from tape for any log disk crash. The added recovery
overhead may be reasonable for small N as the MTTF of the log disks is
still relatively large, but as N increases this will quickly become
unacceptable. In order to improve this reliability while still using simple striping, we have to provide a mirror disk at each of the N disks, which causes problems in the area of cost.

In the next Section, we propose the use of RAID as an extension to striping that has all of the previously described advantages of striping while also providing reliability at lower cost.

3.2.1 Reliable Striping - RAID

As described in Section 2.2.1, RAID extends striping to include parity information in each stripe of data being written. This requires the purchase of at least one additional check disk to allow for the parity information to be added to the stripe. Adding a check disk eliminates the single point of disk failure and brings the reliability of the RAID device to a level higher than that of a single disk. The MTTF of the RAID device is calculated as follows [Patterson88]:

\[
D = \text{total number of disks with data (not including extra check disks)}
\]

\[
G = \text{number of data disks in a group (not including extra check disks)}
\]

\[
C = \text{number of check disks in a group}
\]

\[
\text{ng} = \frac{D}{G} = \text{number of groups}
\]

\[
\text{MTTR} = \text{Mean Time To Repair a failed disk}
\]

\[
\text{MTTF}_{\text{Disk}} = \text{Mean Time To Failure of a single disk}
\]

The MTTF of the entire RAID subsystem with respect to disk failures is given by:

\[
\text{MTTF}_{\text{RAID}} = \frac{(\text{MTTF}_{\text{Disk}})^2}{(D+C*\text{ng})*(G+C-1)*\text{MTTR}}.
\]

The above calculation assumes that failure rates of the disks are exponentially distributed and that failures are independent.
In our case, with a single RAID log device made up of $N$ log disks, assuming we add one check disk for parity information, we have $D = N$, $G = N$, $C = 1$, and $n_G = 1$, giving the reliability of the logging subsystem due to disk failure as:

$$\text{MTTF}_{\text{RAID}} = \frac{(\text{MTTF}_{\text{Disk}})^2}{N(N+1)\text{MTTR}}.$$ 

This reliability is significantly better than the $\text{MTTF}_{\text{Disk}}/N$, which is the MTTF for parallel logging or simple striping without redundancy. As $N$ becomes very large, the RAID system may require multiple check disks to maintain an acceptable level of reliability [Schulze89, Gibson89]. Nevertheless, the cost of adding one or more check disks is significantly less than providing a mirror for each of the $N$ disks. We assume a single check disk throughout this thesis.

As a single logical device, the RAID log maintains the same implementation benefits as simple striping, while maintaining the performance benefits as well. It solves the "hot spot" problem by allowing multiple disks to be written to in parallel, and it also allows for reading the log in parallel during recovery. The overhead of calculating and storing the parity information is very small as long as a full stripe of data can be written with each write to the log device. With a full stripe, the parity can be calculated without any extra disk accesses and written completely in parallel with the data. Assuming Level 5 RAID is being used, the parity calculation is [Patterson88]:

$$\text{new parity} = (\text{old data} \oplus \text{new data}) \oplus \text{old parity}.$$

As long as we are writing a full stripe of data, the parity information is simply calculated as the exclusive OR of all the corresponding data bits in the stripe. To this point, we have been assuming that the log device will always be used at maximum bandwidth. In other words, we either have a full stripe available when a write is required (i.e., at commit), or we delay the write until we have a full stripe (group-commit) [DeWitt84]. However, it may not always be possible to delay the
flush of the log records until the optimal amount of data is available for the log write. The RAID device will have some performance problems with writes of less than a full stripe because of having to read the old data and old parity to calculate the new parity for the stripe. These performance problems were recognized in [Menon93, Seltzer90, Seltzer91, Shirriff94, Stodolsky93, etc.], and are dealt with in the next Section.

3.2.2 Small Writes in a RAID Log: Append-Only RAID

In order to overcome the performance problems with small writes (i.e., less than a full stripe) in a RAID log, we propose a new RAID algorithm we call Append-Only RAID. The purpose of Append-Only RAID is to maintain the low overhead and performance benefits of RAID Level 5\(^1\) while reducing the overhead of small writes to that of a striped disk.

The fundamental assumption in Append-Only RAID is that the device is updated in append-only fashion, i.e., it never experiences random updates. This implies that all writes contain only new data, and there is never any useful old data. Because the old data is always zero, it is no longer required for the parity calculation, eliminating one of the extra disk accesses for small writes. The elimination of old data implies that writes to the log must be a multiple of the RAID block size. Otherwise, the subsequent write to a partially full block would require reading the partially full block as old data in order to calculate the new parity. However, because the log is append-only, the partially full block could easily be cached for the next write, still avoiding a disk access to read the old data (see [Stodolsky93], where the caching of old data is sometimes assumed).

\(^1\)Note that the following discussion does not require Level 5 RAID, but works the same for Level 4 RAID as well. We use Level 5 RAID because it is the most common.
In addition, there is no old parity for the initial write to a stripe and the new parity is simply the exclusive OR of the corresponding data bits in the new data. Append-Only RAID ensures the correctness of the new parity for the initial write to a stripe by zeroing any unused portion of a stripe on the initial write. This is necessary to allow for proper recovery from a hard crash in one of the disks in the device. Subsequent writes to a stripe will require access to the old parity in order to correctly calculate the new parity. By keeping a copy of the last calculated parity in memory, we can avoid any extra disk accesses prior to performing subsequent writes to a stripe as well. Zeroing the remainder of a stripe on the initial write and maintaining the last parity block in memory allows Append-Only RAID to eliminate any extra disk accesses prior to performing a write.

There are other techniques given in the literature for reducing the cost of small writes in a RAID. One technique is called Parity Logging [Stodolsky93]. In this technique, parity updates are cached in memory until a large write can be generated, at which time they are written to a log. At regular intervals, the log is scanned and the parity updates are applied to the standard RAID parity blocks. Although similar to our method in that the updated parity is cached, this method relies on the cache memory being fault tolerant, whereas we have no such restriction because we are writing the parity with each write to the stripe. A second technique, called Floating Parity [Menon93], reserves multiple parity blocks on disk and allows updates to use the parity block closest to the current rotational position of the disk (intending to allow the read/modify/write of the parity to occur in one disk rotation). Although both of these techniques are more general purpose solutions to small writes in a RAID (we require append-only usage), they increase disk overhead and are much more complex solutions. Further, neither technique avoids the disk access to read the
old data unless the old data happens to be cached. Our solution is much simpler and requires no additional disk space and less additional memory to implement, as well as guaranteeing that there is either no old data or the old data is cached.

Since the log is used in append-only fashion, we are able to take advantage of Append-Only RAID when using RAID for the log device. However, there are a couple of issues that must be dealt with when using Append-Only RAID for the log.

The first issue occurs as the current log position wraps around and approaches the oldest page in the log that is still in use (not backed up or still referred to by an open transaction). We obviously won’t overwrite live log data, but if the current logging position is allowed to get closer than one stripe to live log data, our assumptions about there being no old parity for the initial write of a stripe no longer hold. Therefore, the position of the oldest live log data must include the entire stripe containing this old data, preventing the current write position from being within one stripe of live log data. By guaranteeing this, Append-Only RAID’s assumption of no old parity on the initial write to a stripe continues to hold.

There is a second issue that must be dealt in our use of Append-Only RAID for the log concerning our proposal that we rewrite the parity information from the stripe on subsequent writes to a partially full stripe. Rewriting the parity seems to violate a cardinal rule for log devices: after data has been written to disk, it is never over-written until it is no longer needed [Gray91]. The obvious reason for this rule is that if the rewrite fails after partial completion, the previous version is lost forever, which appears to violate the ACID properties required for transaction processing.
To see if we can rewrite the parity in a RAID device and maintain correctness, we must examine three failure cases. First, consider the case where the database system fails with a soft crash prior to either the data or the parity being written to disk. This case is trivial in that the previous parity is still intact on disk. Since the write did not complete, the system reverts to the state it was in prior to the attempted log write, with crash recovery returning the system to a consistent state. Second, consider the case where one of either a data block or the parity block fails to make it to disk. If the data write fails, the parity information can be used to reconstruct the data, and the system continues to operate without loss of information. Similarly, if the parity write fails, the data is still intact and the parity can be reconstructed from the data in the stripe. The normal redundancy inherent in RAID devices allows for these types of failures.

Third, the case where both writes fail must be considered. If both the write of a data block and the write of the parity block fail, a loss of data will occur on the system. In this case, there is no previous parity to reconstruct the data, and without one of the data blocks, it is impossible to reconstruct the parity. The logical conclusion is that the data and parity writes must be serialized to avoid this potential loss of data if we want to rewrite any portion of a stripe, including the parity. In this way, the system knows that one of the writes is complete and can guarantee recoverability if the second write fails. However, the reasonable requirement for the database system is that it must be able to sustain any single failure without loss of data. Multiple device failures must occur before there is a loss of data. Thus rewriting portions of a stripe still meets the dura-
bility requirements placed on the system and allows the algorithm of rewriting the parity in the RAID device to maintain the correctness of the system.\textsuperscript{2}

Our conclusion is that striping and RAID is the best solution for increasing the bandwidth of the log device, providing increased bandwidth and reliability at least cost. Further, our Append-Only RAID algorithm ensures that RAID performs adequately under varying workloads.

\textsuperscript{2}There are published TPC benchmarks [NCR-Informix, HP-Oracle] using RAID for the log that make no apparent provision for writing a full stripe of data for each log write or skipping to the next stripe after a log write, indicating that they are allowing the rewrite of the parity in a RAID stripe while meeting the TPC's durability criterion.
Chapter 4

Log Backup Device Bandwidth

This Chapter discusses the issue of log backup device bandwidth (see Figure 2.1). As with log device bandwidth, the best way to increase the bandwidth of the backup device is to use striping across multiple tape drives. This Chapter compares the use of a backup device with a bandwidth that is less than that of the log device against a more expensive backup device that is the same bandwidth as the log device. It does so by taking each interconnect algorithm from Section 2.3 and comparing configurations using a single tape device versus a striped tape device (see Section 2.2.2) using our criteria and weighting system. We assume that both configurations in a comparison are using the same type of log device, and we do not assume that redundancy is being used in the log backup device. Our results hold regardless of the log device, and our conclusion is that a backup device having the same bandwidth as the log device is always better than a backup device that is less than the bandwidth of the log device. Table 4.1 lists the configurations being compared in this Chapter.

<table>
<thead>
<tr>
<th>Interconnect</th>
<th>Log Device</th>
<th>Log Backup Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Backup</td>
<td>Any log device</td>
<td>Single Tape vs. Striped Tape (Equal to log device)</td>
</tr>
<tr>
<td>Asynchronous Backup</td>
<td>Mirrored (using 3 disks)</td>
<td></td>
</tr>
<tr>
<td>Offline Backup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Staggered Allocation Backup</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Configuration Comparisons for Log Backup Device Bandwidth
4.1 Synchronous Backup: Striped Tape Beats Single Tape

First, we take the synchronous interconnect algorithm for backing up the log device. Figure 2.4 in Section 2.3 illustrates the synchronous backup interconnect.

Using our criteria, we first compare the two for whether either will lose data for any single failure. Since we are backing up the log synchronously, no single crash will cause us to lose data. All log data will be either on the log device or the log backup device following any single failure. This is the same for any log backup device, so both configurations are considered Excellent for this criterion.

The next criterion is maximum average speed. When using synchronous backup, the maximum average speed will be the slower of either the log device or log backup device. In our model, this is the speed of the backup device. The maximum average speed for the striped tape device is the same as the log device so it is classified as Excellent for this criterion and is declared the winner. Single tape is classified as Poor because it will never be greater than 33% of the speed of the log device for any of the log devices in our model.

Similarly, the striped tape configuration wins for the criterion of maximum burst speed. With synchronous backup, the maximum burst speed is the same as the maximum average speed, so the arguments for maximum average speed hold here as well. The striped tape configuration is again classified as Excellent, with the single tape configuration classified as Poor.

For the criterion of cost/performance, however, the single tape configuration wins. Both configurations require two log backup devices due to the synchronous interconnect algorithm. But the striped tape configuration requires the purchase of a software package to implement the tape striping as well as the purchase of more tape drives. Table 4.2
shows the performance per dollar for synchronous backup as we add tape drives to the backup device. After the second tape drive is added to begin striping, we obtain better performance per dollar with each added drive. The second drive requires the purchase of the striping software, so the performance per dollar decreases at that point. Because the crossover point is well beyond any reasonable bandwidth requirements for the log backup device, single tape is declared the winner.

<table>
<thead>
<tr>
<th>Bandwidth (M/second)</th>
<th>Striping Software</th>
<th>Tapes (x2) $3000 each</th>
<th>Total Cost</th>
<th>$/M/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>0</td>
<td>2</td>
<td>6000</td>
<td>12000</td>
</tr>
<tr>
<td>1</td>
<td>10000</td>
<td>4</td>
<td>22000</td>
<td>22000</td>
</tr>
<tr>
<td>2</td>
<td>10000</td>
<td>8</td>
<td>34000</td>
<td>17000</td>
</tr>
<tr>
<td>3</td>
<td>10000</td>
<td>12</td>
<td>46000</td>
<td>15333</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>16</td>
<td>58000</td>
<td>14500</td>
</tr>
</tbody>
</table>

Table 4.2: Synchronous Interconnect Performance Per Dollar

The next criterion is recovery time for a soft crash. Soft crash recovery will ignore the backup device and use the given log device to process log records and bring the system back into a consistent state. Since both configurations are using the same type of log device, they are equal for this criterion. Because we have not specified the type of the log device, we do not attempt to classify the configurations for a soft crash.

Hard crash recovery is next and it favors the striped tape backup device. For a hard crash in a data disk or log disk (without redundancy), the striped tape device is able to read back the data at a rate that is equal to the bandwidth of the log device. The single tape device restores the log data at a rate that is much less than that of the log device. For the hard crash of a tape drive, we consider a striped tape backup as better than a single tape backup for the crash of a tape drive.
because it can continue to backup log data after the crash. Further, the single tape configuration may require an archive of the database after the drive is replaced to allow synchronous backup to begin again (see Section 2.4.6). Both are equal for the crash of a magnetic tape as each may require an archive of the system. Overall, the striped tape configuration is considered Average for hard crash recovery because it is able to restore log data at a rate that is equal to the bandwidth of the log device and requires no archive for the crash of a tape drive. The single tape configuration is classified as Poor and is the loser for this criterion.

Finally, there are the criteria of hardware and software changes required to support the configurations. Neither configuration requires hardware changes with respect to the log backup device, so we consider both configurations as Excellent and equal for this criterion. For software changes, the striped tape device requires software changes to implement, so it is rated as Average. The single tape configuration uses no special software, so it is rated as Excellent and wins for this criterion.

We conclude that synchronous backup to a single tape wins in terms of cost and the need for software changes, but synchronous backup to striped tape wins for speed criteria and hard crash recovery, with the
rest of the criteria rated as equal. Based on our weighting system, the striped tape configuration is the overall winner in the comparison. Table 4.3 shows the results for this comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Single Tape</th>
<th>Striped Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes.</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td>28</td>
<td>35</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.25</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.3: Synchronous Backup: Single Tape vs. Striped Tape

4.2 Asynchronous Backup: Striped Tape Beats Single Tape

Next we examine the use of any log device with asynchronous backup to a single tape as opposed to asynchronous backup to a striped tape device. Figure 2.3 in Section 2.3 illustrates the asynchronous interconnect algorithm.

Once again we start with loss of data for any crash. Whether or not we lose data for any single crash depends on the type of log device used, and is the same regardless of the log backup device. If there is no redundancy in the log device, a single hard crash in a log device will result in a loss of data to the system. The asynchronous backup is always at least a short distance behind the current log write position as it attempts to read the log disk and backup the log data. The amount of data lost therefore, is the difference between the current
log backup position and the current log write position. Although the striped tape backup will be closer to the current log write position at the time of a crash because its bandwidth is equal to the log device, we do not classify one configuration as better than another based on the fact that it loses less data. Because both configurations are using the same type of log device, we rate them as equal for this criterion. We will assume that redundancy is built into the log device and rate them both as Excellent for this criterion.

Next, consider the maximum average speed of the logging subsystem. As indicated in Section 2.3 and demonstrated in Chapter 6, the extra seeks in the log device for asynchronous backup of the log can have a significant impact on the speed of the logging subsystem. The closer the speed of the log backup to the speed of the logging device, the smaller the impact to performance. Since the striped tape device has the same bandwidth as the log device, the seek distances are minimized. The single tape device is at most 33% of the speed of the log device, so the seeks are much worse in this case, resulting in a greater degradation in performance. Therefore, the striped tape configuration wins for this criterion. Our experimental results indicate a 10 to 15% slowdown using asynchronous backup in spite of the backup device bandwidth being equal to the log device bandwidth (see Chapter 6), so the striped tape configuration is rated as Average for this criterion. Because the single tape configuration can obtain at most a maximum average speed that is 33% of the speed of the log device, it is rated as Poor.

The maximum burst speed for both configurations has the same characteristics and is similarly influenced by seek time. Our experimental results indicate that the striped tape configuration will decrease maximum burst speed by 10 to 15% (see Chapter 6), so it is rated as Average. The single tape configuration will reduce maximum burst speed by 30 to 40% (see Chapter 6), so it is rated as Poor for this criterion.
For the criterion of cost/performance, the single tape configuration wins for the same reason it won for synchronous backup. Even though the lower bandwidth device decreases the maximum burst speed much more than the higher bandwidth one, the high cost of moving from a single tape backup to a striped tape backup allows the single tape to remain the winner in the comparison. After the second tape is added to the device, each subsequent addition reduces the cost per performance for the log backup.

The criteria for recovery time for soft and hard crashes, and special hardware and software changes follows exactly as they did above for synchronous backup. The single tape configuration wins for cost/performance and not needing any special software, the striped tape configuration wins for recovery time for a hard crash, and both are equal for recovery time for a soft crash and the need for special hardware.

We conclude that as with synchronous backup, asynchronous backup to a striped tape device wins out over asynchronous backup to a single tape. Table 4.4 contains the overall results for this comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Single Tape</th>
<th>Striped Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes.</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td>28</td>
<td>35</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.25</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.4: Asynchronous Backup: Single Tape vs. Striped Tape
4.3 Offline Backup: Striped Tape Beats Single Tape

We now examine the use of any log device with offline backup to a single tape versus offline backup to a striped tape. Figure 2.5 in Section 2.3 illustrates the offline interconnect algorithm.

Beginning with loss of data for any single crash, offline backup follows the same argument as asynchronous backup. For Offline interconnect, loss of data for a hard crash depends completely on whether or not there is redundancy in the log device. Offline backup is never backing up the current log device, so if the current device crashes there will be a loss of data unless it provides its own redundancy. If the previous log device crashes before all of the log data has been backed up, then this will also result in a loss of log data unless the device is able to recover from the crash itself. We will again assume that redundancy is inherent in the log device, and consider both configurations to be equal and Excellent for this criterion.

The maximum average speed when using offline backup will always be the slower of either the log device or log backup device. Due to the time it takes to change tapes in the backup device (see Section 2.2.2), we are unable to maintain a maximum average speed that is the same as the maximum average speed of the log device even if the backup device speed is equal to the maximum average speed of the log device. Therefore, the striped tape configuration is considered Average for this criterion. On the other hand, the single tape device is at most 33% of the speed of the log device, making the maximum average speed at most 33% of the log device maximum average speed, so the single tape configuration is considered Poor for this criterion, with the striped tape configuration the winner.

When using offline backup, the maximum burst speed is unaffected by the log backup device or the interconnect algorithm between the log device and the log backup device. This is because there is no inter-
ference with the logging process by the log backup process. Both the single and striped tape configurations are considered Excellent for this criterion and are equal.

The criteria for cost/performance, recovery time for soft and hard crashes, and special hardware and software changes required follows exactly as it did above for both synchronous and asynchronous backup.

Although the fact that the two configurations are equal in terms of maximum burst speed makes this comparison much closer than the other two, we still conclude that offline backup from any log device to striped tape wins over offline backup to a single tape device. Table 4.5 contains the overall results for this comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Single Tape</th>
<th>Striped Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes.</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td>33</td>
<td>35</td>
</tr>
</tbody>
</table>

Table 4.5: Offline Backup: Single Tape vs. Striped Tape

4.4 Staggered Allocation Backup: Striped Tape Beats Single Tape

We finally examine the use of a mirrored log device using three disks with staggered allocation backup to a single tape versus staggered allocation backup to a striped tape. Figure 2.6 in Section 2.3 illustrates the staggered allocation interconnect algorithm.
Beginning with loss of data for any single crash, staggered allocation backup will never lose data for any single crash because mirroring is being used in the log device. We consider both configurations to be equal and Excellent for this criterion.

Because staggered allocation is a variation on offline backup, the maximum average speed when using staggered allocation will always be the slower of either the log device or log backup device. Similar to the discussion of offline backup above, the striped tape configuration is considered Average for this criterion due to the time it takes to change tapes, while the single tape configuration is considered Poor because it is at most 33% of the speed of the log device. Thus the striped tape configuration is the winner for this criterion.

As with offline backup, the maximum burst speed is unaffected by the log backup device or the interconnect algorithm between the log device and the log backup device. This is because there is no interference with the logging process by the log backup process. Both the single and striped tape configurations are considered Excellent for this criterion and are equal.

The criterion of hard crash recovery performance is similar to the comparisons using the other three interconnect algorithms. The only exception is that we do not require recovery for the hard crash of a log disk for either configuration because of mirroring in the log device. However, the striped tape configuration is still the winner for this criterion because it is able to continue backing up log data after the crash of a tape drive.

The criteria for cost/performance, recovery time for a soft crash, and special hardware and software changes required follows exactly as it did above for the other three interconnect algorithms.
The results of this comparison are exactly the same as for offline backup, with the striped tape configuration declared the winner. The overall results are shown in Table 4.6.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Single Tape</th>
<th>Striped Tape</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes.</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Totals</td>
<td>33</td>
<td>35</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.06</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Staggered Allocation Backup: Single Tape vs. Striped Tape

4.5 Conclusions

We have taken each log backup interconnect algorithm and compared a configuration using a single tape device as the log backup device versus a configuration using a striped tape device that is the same bandwidth as the log device. In each comparison, the striped tape configuration was declared the winner. Thus we conclude that for any log device and any log backup method, backing up the log to a tape device that has the same bandwidth as the log device always wins over backing up the log to a tape device whose bandwidth is less than the log device.
Chapter 5

The Interconnect - Evaluation of Log Backup Methods

This Chapter examines the issue of how best to interconnect the log device and the log backup device, otherwise known as the log backup method (see Figure 2.1). Our claim is that synchronous backup of the log provides the best performance at the least cost. We attempt to prove this claim by comparing various logging subsystem configurations whose principle difference is in the area of the interconnect algorithm. We examine configurations for three cases: low bandwidth (250K/second), medium bandwidth (1.5M/second), and high bandwidth (5M/second). We will make use of redundancy in the backup device as well as in the log device (where necessary). In each case, we compare three of the interconnect algorithms (synchronous, asynchronous, and offline) using our criteria and weighting system to determine the best of the three. For the offline interconnect algorithm, we will substitute the improved staggered allocation backup method whenever mirroring is to be used in the log device.

Note that our discussion of the log backup interconnect does not deal with the impact of transaction rollback on the logging rate of the system. Rollback should magnify the disk contention problems already inherent in asynchronous backup as a transaction attempts to read its log records in reverse, possibly requiring disk accesses to do so. Rollback will have the same impact on both offline and synchronous backup as it adds a small amount of contention for the log device to
each of these interconnect algorithms. Because rollbacks should be infrequent and will not significantly impact our results, for the sake of simplicity we ignore the issue of transaction rollback.

5.1 Low Bandwidth

The first case to examine is low bandwidth. Examining configurations that only require 250K/second for the log rate basically removes performance as a determining factor in which is the better configuration and places more emphasis on cost. Our conclusion for this very low bandwidth is that synchronous interconnect is the best method to use as it provides the required performance at the least cost. The following is our analysis.

5.1.1 Synchronous Interconnect Beats Asynchronous Interconnect

Our first comparison is between synchronous interconnect and asynchronous interconnect. The hardware and software costs for these two configurations are given in Table 5.1. For low bandwidth, synchronous interconnect consists of a single log disk, two pairs of mirrored tapes, and the tape mirroring software package. It does not require a second log disk for the mirror because the log backup serves as a mirror of the log disk. The two mirrored pairs of tape drives is due to syn-
chronous backup requiring two log backup devices. For asynchronous backup, the configuration consists of a mirrored disk, one mirrored tape drive and the software to allow mirroring in the backup tapes.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Disks</td>
<td>$10,000</td>
<td>1 $10,000</td>
<td>2 $20,000</td>
</tr>
<tr>
<td>Tape Mirroring Software</td>
<td>$10,000</td>
<td>1 $10,000</td>
<td>1 $10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>4 $12,000</td>
<td>2 $6,000</td>
</tr>
<tr>
<td>Total Cost</td>
<td>$32,000</td>
<td>$36,000</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1: Low Bandwidth: Synchronous vs. Asynchronous Cost

The first criterion is lost data for any single failure. Since both configurations are using the WAL protocol, neither will lose data as the result of a soft crash. Synchronous backup will not lose data for any single hard crash because of the log backup acting as a mirror of the log disk. Asynchronous backup depends on the mirrored log disks to prevent any single hard crash from causing a loss of data. Therefore, neither configuration will lose data for any single failure, making both configurations equal and classified as Excellent for this criterion.

The second criterion is maximum average speed. Because the bandwidth of both the log device and the log backup device is much greater than the required 250K/second of log system bandwidth, both configurations are able to operate at a maximum average speed of greater than or equal to 250K/second. Thus both configurations are rated as Excellent and are considered equal.

Similarly, maximum burst speed is also equal for the two configurations. Since 250K/second is all the speed that is required, both configurations are able to exceed this maximum burst speed. As such, each is rated as Excellent and equal for this criterion.
Next is the criterion of cost/performance. As shown in Table 5.1, the cost in dollars for synchronous backup is less than the cost for asynchronous backup. Note that synchronous backup requires two extra tape drives, while asynchronous backup requires an extra disk. Thus, synchronous backup will be less expensive as long as two tape drives cost less than one disk. Since both provide the same effective speed for the low bandwidth requirements, synchronous backup provides better speed per cost and is the winner for the criterion of cost/performance.

Recovery time for a soft crash is next. Recovery time reading the log for a soft crash is completely dependent on the bandwidth of the log device. Since both configurations are using a single or mirrored log disk, both have the same bandwidth in their log devices. They are both rated as Average for this criterion and are considered equal.

For recovery time for a hard crash, asynchronous backup is the winner in the comparison. Both configurations are equal for the crash of a data disk because both are using the same type of log backup device (assuming no redundancy in the data disk). However, neither is able to read back the log at a rate that is equal to the bandwidth of the log device. The redundancy in the log device allows the asynchronous backup to need no recovery for a hard crash in a log disk. Synchronous backup must recover from tape in this case, making asynchronous better for this type of crash. The two are equal for the hard crash of a tape drive or the hard crash of a magnetic tape as neither requires any recovery due to the use of mirroring. Overall, both are considered Poor for this type of crash because they are unable to read back log data at a rate equal to the maximum bandwidth of a log device. Because asynchronous backup needs no recovery for the crash of a log disk, we declare it to be the winner for this criterion.
Both configurations are rated as Excellent and equal for the criterion of special hardware changes because neither requires any special hardware. With respect to special software changes, they are both considered to be Average because they require the purchase of a software package to allow mirroring of tape drives.

Our conclusion is that these two configurations are rated as equal with the exception of the criteria of cost/performance and hard crash recovery. Synchronous interconnect wins in the area of cost/performance as long as two tape drives can be purchased for less than the cost of one disk drive, while asynchronous interconnect wins for hard crash recovery. Because we give cost/performance more weight than hard crash recovery, synchronous interconnect is declared the winner. The results of this comparison are shown in Table 5.2.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.06</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.2: Low Bandwidth: Synchronous vs. Asynchronous Results
5.1.2 Synchronous Interconnect Beats Offline Interconnect

The second comparison for low bandwidth is synchronous interconnect and offline interconnect. For synchronous interconnect, the configuration is the same as in the previous comparison. For offline interconnect, we will use staggered allocation backup as the better offline interconnect configuration given in [Gray91] and described in Section 2.3. Thus the offline backup requires three log disks, two mirrored tape drives and the software to allow mirroring in the backup tapes. The configurations and their costs are summarized in Table 5.3.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Synchronous</th>
<th>Offline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Qty</td>
<td>Cost</td>
</tr>
<tr>
<td>Log Disks</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Tape Mirroring Software</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>4</td>
<td>$12,000</td>
</tr>
<tr>
<td>Total Cost</td>
<td></td>
<td></td>
<td>$32,000</td>
</tr>
</tbody>
</table>

Table 5.3: Low Bandwidth: Synchronous vs. Offline Cost

The comparison between these two configurations follows much the same as the one above. Beginning with loss of data for any single failure, both configurations are again rated as Excellent and neither will lose data. The reasons for synchronous backup not losing data were described above. Offline backup will not lose data for any hard crash because it is using mirroring in the log device. Therefore, both configurations are equal for the criterion of lost data.

The configurations are also equal in terms of maximum average speed. The log disk and backup tape bandwidths are much larger than that required for low bandwidth, so both configurations are able to maintain
the required maximum average speed. Similarly, maximum burst speed is also the same for the configurations as each is easily able to handle the low bandwidth requirements of the system.

The criterion of cost/performance is again the difference in the comparison. As shown in Table 5.3, synchronous backup is the least expensive of the two configurations. Since the two are equal in terms of the speed criteria, synchronous backup provides better performance per dollar and is the winner in the area of cost/performance. Synchronous backup uses 2 more tape drives and offline backup uses 2 more disks, thus our conclusion holds as long as the price of a tape drive is less than the price of a log disk.

Recovery time for a soft crash follows exactly as it did for the previous comparison, with both being rated as Average. For recovery time for a hard crash, the offline backup is the winner for the same reasons as shown in the previous comparison between synchronous and asynchronous backup.

Both configurations are rated as Excellent for the criterion of special hardware changes because neither requires special hardware. They are both rated as Average for the criterion of special software changes and are equal for this category as well.
Thus we conclude that synchronous backup wins in the area of cost/performance and offline wins in the area of hard crash recovery, with all other criteria being equal. Once again, synchronous interconnect is declared the winner. Table 5.4 shows the results of this comparison.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Offline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.06</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.4: Low Bandwidth: Synchronous vs. Offline Results

Our conclusion for low bandwidth is that synchronous backup is the better interconnect algorithm as long as the price of a tape drive is no more than half the price of a disk drive for disk and tape drives providing the required bandwidth. Otherwise, asynchronous backup appears to provide the best performance per dollar at low bandwidth.

5.2 Medium Bandwidth

The second case to examine is medium bandwidth, where medium bandwidth is a desired speed of 1.5M/second, which is the speed of a single disk in our model. We perform the same comparisons as above for low bandwidth, comparing synchronous interconnect against both asynchronous and offline interconnect. Our conclusion is that synchronous interconnect is again the winner.
5.2.1 Synchronous Interconnect Beats Asynchronous Interconnect

We begin by comparing synchronous interconnect against asynchronous interconnect. For a speed of 1.5M/second, synchronous interconnect still requires a single log disk. In order to match the bandwidth of the log disk, the bandwidth of 3 tape drives is required. To provide redundancy in the log backup device, we will use RAID Level 4 and 4 tape drives for a single backup device. Because synchronous backup requires two log backup devices, it requires a total of 8 tape drives as well as the software package to allow for tape striping. Asynchronous backup requires a mirrored pair of log disks, a single log backup device consisting of 4 tape drives, and the software package to allow RAID Level 4 in the log backup device. The summary of the hardware and software requirements and their costs for this comparison are contained in Table 5.5.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Qty Synchronous</th>
<th>Qty Asynchronous</th>
<th>Cost Synchronous</th>
<th>Cost Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Disks</td>
<td>$10,000</td>
<td>1</td>
<td>2</td>
<td>$10,000</td>
<td>$20,000</td>
</tr>
<tr>
<td>Tape Striping Software</td>
<td>$10,000</td>
<td>1</td>
<td>1</td>
<td>$10,000</td>
<td>$10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>8</td>
<td>4</td>
<td>$24,000</td>
<td>$12,000</td>
</tr>
<tr>
<td>Total Cost</td>
<td></td>
<td>$44,000</td>
<td>$42,000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5.5: Medium Bandwidth: Synchronous vs. Asynchronous Cost

The criterion of loss of data for any single failure follows exactly as above. Synchronous backup mirrors the log device with the log backup device and asynchronous backup uses mirroring in the log device. Both configurations are classified as Excellent and equal for this criterion.
For the criterion of maximum average speed, synchronous backup is the winner. Synchronous backup maintains a maximum average speed of the slower of either the log device or log backup device. In this case, they are the same bandwidth, so the maximum average speed is the same as the bandwidth of the log device, making it Excellent for this criterion. Note that the required down time to change tapes in the backup device does not affect synchronous backup because it is using two log backup devices. Asynchronous backup is unable to utilize the full bandwidth of the log backup device due to the contention with the log device. Our experimental results (see Chapter 6) indicate that the maximum average speed is reduced by about 10 to 15% from the maximum speed of the log device when the log backup bandwidth is the same as the log device bandwidth. This is without taking into account the lost time when changing tapes in the backup device. Asynchronous backup is therefore Average for maximum average speed and synchronous backup is the winner.

Maximum burst speed is next. With synchronous backup, the maximum burst speed is the same as the maximum average speed, so it is again Excellent for this criterion. Asynchronous backup faces the same slowdown in maximum burst speed as it did for maximum average speed, making it Average for this criterion with synchronous backup the winner.

Cost/performance is the next criterion. Synchronous backup provides 1.5M/second for both the maximum average and burst speeds for a cost of $44,000. Assuming that the slowdown in maximum average and burst speeds is 10% for asynchronous backup, it provides 1.35M/second for a cost of $42,000. This evaluates to a cost/performance of $29,333 per M/second for synchronous backup, and $31,111 per M/second for asynchronous backup. This means that synchronous backup provides better performance per dollar and is the winner in terms of cost/performance.
The criteria for soft crash recovery, hard crash recovery, special hardware changes, and special software changes follows similar to our above comparison of synchronous and asynchronous interconnect at low bandwidth. Asynchronous backup wins in terms of hard crash recovery with both configurations rated as equal for the rest of these criteria. Asynchronous backup is now considered Excellent for hard crash recovery because it is able to read back log data at a rate equal to the bandwidth of the log device. Synchronous backup is now considered Average for hard crash recovery for the same reason.

Our conclusion is that for medium bandwidth, synchronous interconnect is better than asynchronous interconnect. Synchronous interconnect wins because it provides better performance at least cost, while asynchronous interconnect is only better for the criterion of hard crash recovery. The results of this comparison are shown in Table 5.6.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>26</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.46</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.6: Medium Bandwidth: Synchronous vs. Asynchronous Results
5.2.2 Synchronous Interconnect Beats Offline Interconnect

Next we compare synchronous interconnect against offline interconnect for the medium bandwidth case. The configuration for synchronous interconnect is exactly as it was for the medium bandwidth case above. For offline interconnect, we will again use the improved staggered allocation method with a log device configuration of 3 log disks. We will also require one log backup device, which is made up of 4 tape drives using Level 4 RAID with 3 drives for log backup data and one for a parity drive, along with the software package for implementing RAID Level 4 in the log backup device. The summary of the configurations and their cost is contained in Table 5.7.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Synchronous</th>
<th>Offline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Qty</td>
<td>Cost</td>
</tr>
<tr>
<td>Log Disks</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Tape Striping Software</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>8</td>
<td>$24,000</td>
</tr>
<tr>
<td><strong>Total Cost</strong></td>
<td></td>
<td><strong>$44,000</strong></td>
<td><strong>$52,000</strong></td>
</tr>
</tbody>
</table>

Table 5.7: Medium Bandwidth: Synchronous vs. Offline Cost

The first criterion of lost data is the same as it was for the low bandwidth comparison between these two interconnect algorithms. Each is classified as Excellent and the two are equal.

For the criterion of maximum average speed, synchronous backup is the winner. As indicated above, synchronous backup maximum average speed is the same as the log device bandwidth, which is 1.5M/second, making it Excellent for this criterion. Offline interconnect has no contention in the log device by the log backup process, so it should be able to maintain a maximum average speed equal to the bandwidth of the log device. However, it is unable to do so due to the interference in the
log backup caused by changing tapes in the log backup device. Assuming
that the tape drives allow ejecting the tape without rewinding, it
still takes at least 1 minute on average to change the tapes in the
device (see Section 2.2.2). Therefore offline backup will fall behind
90M each time the tapes are changed in the device (60 seconds * 1.5M/
second log rate). It will have to make this up after each tape change
prior to the next log device switch. There are two possible ways to
accomplish this. One is to increase the bandwidth of both the log device
and the log backup device such that they are actually enough faster
than the required log rate of 1.5M/second to allow the log backup to
overcome the deficit caused by tape switches. The second and lower cost
approach is to add a second log backup device to the offline backup
configuration and eliminate the tape switch problem altogether. This
second option obviously adds to the cost of the configuration. Note
that if neither option is chosen, the offline backup will fall further
and further behind. The system will eventually either be forced to halt
until some log space can be backed up and made available, or it will
degenerate into asynchronous backup by allowing the log backup and the
current log to be on the same disk. If we assume that a second log
backup device is added to allow offline interconnect to reach the
required speed, it is rated as Excellent for this criterion. This makes
the two configurations equal in terms of maximum average speed.

Next is the criterion of maximum burst speed. Synchronous intercon­
nect maximum burst speed is the same as the maximum average speed and
is Excellent for this criterion. Offline interconnect maximum burst
speed is always the maximum burst speed of the log device, making it
Excellent for this criterion and the two configurations equal for max­
imum burst speed.
Cost/performance is the next criterion, with synchronous interconnect the winner. Synchronous interconnect provides at least as much speed as offline interconnect and at reduced cost. This is magnified if offline interconnect adds a second log backup device to the configuration. Even without this second log backup device, offline interconnect is higher in cost as long as a log backup device costs less than two log devices of equivalent bandwidth. Adding a second log backup device to offline interconnect means that it will never cost less than synchronous interconnect.

The criteria for soft crash recovery, hard crash recovery, special hardware changes, and special software changes follows similar to our above comparison of synchronous and offline interconnect at low bandwidth. Offline backup wins in terms of hard crash recovery and is now considered Excellent because it can restore log data at the maximum bandwidth of the log device, with synchronous backup now considered Average for the same reason. Both configurations are rated as equal for the rest of these criteria.

Thus we conclude that synchronous interconnect provides at least the performance of offline interconnect and at reduced cost, making it the better choice for log backup at a medium bandwidth log rate. Table 5.8 shows the results of this comparison.

The only way that offline interconnect can win this comparison is if it can match the speed of the synchronous interconnect while reducing its cost to be less than or equal to the synchronous interconnect configuration. If we assume that we can obtain the required increase in bandwidth by adding the bandwidth of one disk to the log device and one tape to the backup device of the offline configuration, we must add three disks and one tape to the configuration (using striping in the log device). This means that synchronous backup is using one disk, eight tapes, and the tape striping software, while the offline backup
is using six disks and five tapes, along with the striping software. Synchronous backup is using three more tapes and offline backup is using five more disks which are also more expensive due to striping. For offline backup to win in terms of cost/performance, these five disks must cost less than the three tapes.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Offline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>38</td>
<td>36</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.06</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.8: Medium Bandwidth: Synchronous vs. Offline Results

Our conclusion at medium bandwidth is the same as low bandwidth: synchronous interconnect is the best interconnect algorithm. It always performs at or above the level of the other two interconnect algorithms and does so for lower cost.

5.3 High Bandwidth

The final case to be examined is for high bandwidth in the log device, with high bandwidth being a 5M/second log rate. We perform the same interconnect comparisons as we did above and again conclude that synchronous backup is the winner.
5.3.1 Synchronous Interconnect Beats Asynchronous Interconnect

We again begin with a comparison between synchronous interconnect and asynchronous interconnect. To provide for a 5M/second log rate, we will require the use of striping in the log device and the bandwidth of 4 log disks. Although synchronous backup is mirroring the log device with the backup device, we will want to use RAID Level 5 in the log device due to the decreased MTTF that having 4 log disks causes. For example, assume an MTTF of 30,000 hours for a single disk (see [Patterson88, Shulze89, Mourad92]). The MTTF of 4 of these disks is 30,000/4 = 7500 hours, or about 10 months. Adding a single check disk and using RAID significantly increases the MTTF. Recall that the MTTF of a RAID device is calculated as [Patterson88]:

\[ MTTF_{RAID} = \frac{(MTTF_{Disk})^2}{(D+C*n_G)*(G+C-1)*MTTR)} \]

In our case, \( D = 4 \), \( G = 4 \), \( C = 1 \), \( n_G = 1 \). Assuming a MTTR of 72 hours and \( MTTF_{Disk} \) of 30,000 hours, we have \( MTTF_{RAID} = (30,000)^2 / (5 * 4 * 72) \), which is 625,000 hours (approximately 71 years), far exceeding the useful lifetime of the disks. Therefore, the cost for the extra check disk is reasonable given the extra reliability it provides, so RAID will be used in the synchronous backup configuration.
In order to attain a bandwidth of 5M/second in the log backup device, we require striping the backup across 10 tape drives, and add one more drive to allow for Level 4 RAID in the backup device. Since synchronous backup requires two log backup devices, it requires 22 tape drives as well as the tape striping software. The asynchronous interconnect configuration requires a single RAID for the log device and a single striped log backup device along with the tape striping software. The summary of the configurations and their associated costs is shown in Table 5.9.

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Synchronous Qty</th>
<th>Cost</th>
<th>Asynchronous Qty</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Disks (RAID)</td>
<td>$12,000</td>
<td>5</td>
<td>$60,000</td>
<td>5</td>
<td>$60,000</td>
</tr>
<tr>
<td>Tape Striping Software</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>22</td>
<td>$66,000</td>
<td>11</td>
<td>$33,000</td>
</tr>
<tr>
<td>Total Cost</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$136,000</td>
</tr>
</tbody>
</table>

Table 5.9: High Bandwidth: Synchronous vs. Asynchronous Cost

For the criterion of lost data for any failure, neither of these configurations will ever lose data, so both are rated as Excellent for this criterion. For the criterion of maximum average speed, the argument follows exactly as it did for the medium bandwidth comparison between these two configurations: asynchronous backup slows the maximum average speed by 10 to 15%, while synchronous backup is able to take advantage of the full bandwidth of the log device. Asynchronous backup still interferes with the log disk and hurts performance. This does not take into account the cost for changing tapes in the backup device with asynchronous backup, which will significantly affect maximum average speed at this high bandwidth. If it takes 1 minute to change tapes, the log backup has fallen behind a further 300M of new log data (60 seconds
* 5M/second). Again, we can add a second log backup device to asynchronous backup, making their costs equal, but we will still lose the 10 to 15% performance due to the contention in the log device.

The maximum burst speed criterion is also won by synchronous backup. It is reduced by the same 10 to 15% as maximum average speed in asynchronous backup. Further, without a second log backup device, asynchronous backup will damage maximum burst speed even more because this lost backup time due to changing tapes causes the backup to be further behind the current log position, creating longer seeks in the log device.

For the criterion of cost/performance, the two methods are roughly equal. Synchronous backup is able to use the full bandwidth of the log device for both maximum average and burst speeds, giving it 5M/second for $136,000, or $27,200 per M/second. If we assume that both maximum average and burst speeds are cut by an average of 25% when using asynchronous backup, this gives 3.75M/second for $103,000, or $27,466 per M/second. A cut of 25% is actually optimistic since tape switching will actually reduce the effective bandwidth of the log backup device and therefore hurt maximum burst speed by closer to 30 to 40% or more (see Chapter 6). Adding a second backup device to the asynchronous backup configuration makes their overall costs equal, but asynchronous interconnect would lose in that case because it maintains a 10 to 15% performance penalty. Although allowing for only a 25% drop in performance for asynchronous backup is optimistic, we rate their cost/performance as equal.

The rest of the criteria are all equal for these two configurations. Each has the same type of log device and log backup device, so each performs the same for crash recovery. They also each require special hardware for the RAID disks and special software for tape striping.
Thus we conclude that for high bandwidth, synchronous interconnect is better than asynchronous interconnect, principally due to its performance benefits. The results of this comparison are shown in Table 5.10.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>30</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.33</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.10: High Bandwidth: Synchronous vs. Asynchronous Results

5.3.2 Synchronous Interconnect Beats Offline Interconnect

Our final comparison is between synchronous interconnect and offline interconnect for high bandwidth. The synchronous configuration is the same as described above for high bandwidth. For the offline configuration, we will not multiply the staggered allocation model we have been using by 4 and use simple striping to obtain the required bandwidth. Instead, we will use two RAID log devices because this will reduce the cost. Using striping with the staggered allocation model requires us to purchase 3 sets of 4 striped disks, for a total of 12 disks. The overhead for redundancy in this model is 4 disks, while the RAID configuration only requires the purchase of 10 disks with each RAID having
a single check disk for a redundancy overhead of 2. The offline configuration also requires the purchase of a single log backup device of 11 tape drives along with the software package to allow RAID Level 4 in the log backup device. Table 5.11 shows the configurations and their associated costs.

Table 5.11: High Bandwidth: Synchronous vs. Offline Cost

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost Each</th>
<th>Synchronous Qty</th>
<th>Cost</th>
<th>Offline Qty</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log Disks (RAID)</td>
<td>$12,000</td>
<td>5</td>
<td>$60,000</td>
<td>10</td>
<td>$120,000</td>
</tr>
<tr>
<td>Tape Striping Software</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
<td>1</td>
<td>$10,000</td>
</tr>
<tr>
<td>Log Backup Tape Drives</td>
<td>$3,000</td>
<td>22</td>
<td>$66,000</td>
<td>11</td>
<td>$33,000</td>
</tr>
<tr>
<td>Total Cost</td>
<td></td>
<td></td>
<td>$136,000</td>
<td></td>
<td>$163,000</td>
</tr>
</tbody>
</table>

Similar to the previous comparisons, these configurations will also not lose data for any single failure. For the criterion of maximum average speed, offline backup faces the problem of changing tapes in the same way as for medium bandwidth. Assuming it takes one minute to change tapes, the backup will be 300M behind after the change (60 seconds * 5M/second). The maximum bandwidth of the log device is 6M/second, so we will assume that this extra 1M/second is enough to allow it to make up the lost time. However, the backup device must have two tapes added to it to allow it to make up the lost time as well. Assuming that the extra tapes are added to the offline configuration, both configurations are then classified as Excellent and are equal.

For maximum burst speed, both configurations are equal as they both provide a maximum burst speed equal to the maximum bandwidth of the log device. In terms of cost/performance, synchronous backup is the winner. It provides maximum average and burst speeds that are equal to the maximum bandwidth of the log device, and it does so for lower cost than offline backup. After adding two more tape drives to the offline con-
figuration to make up for the lost average speed, synchronous backup
is using nine more tape drives while offline backup is using five more
disk drives. The only way that offline backup could win in terms of
cost is if the nine tapes cost more than the five disks, which is not
the case in our model.

The rest of the criteria are all equal for these two configurations
for the same reasons as synchronous interconnect and asynchronous
interconnect were equal for these criteria.

Our conclusion is again that synchronous interconnect wins over
offline interconnect and is the best interconnect algorithm for high
bandwidth. The results of this comparison are summarized in Table 5.12.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Synchronous</th>
<th>Offline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loss of data for any single failure.</td>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>Maximum average speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Maximum burst speed.</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Cost/performance.</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>Recovery time for a soft crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Recovery time for a hard crash.</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Special hardware changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Special software changes required.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>40</td>
<td>36</td>
</tr>
<tr>
<td>Ratio</td>
<td>1.11</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.12: High Bandwidth: Synchronous vs. Offline Results

5.4 Conclusions

Having examined three cases, low, medium, and high bandwidth, we con-
clude that in each case synchronous interconnect is the best intercon-
nect algorithm, providing the best performance at the least cost. Our
results hold in the low bandwidth case as long as a tape drive can be
purchased for half the cost of a disk drive for the bandwidth required.
The results hold in the medium bandwidth case as long as a tape drive does not cost significantly more than a disk drive, and they hold in the high bandwidth case as long as a striped tape array does not cost more than the equivalent bandwidth striped disk array.
Chapter 6

Results from an Implemented Prototype

This Chapter describes the results of an implemented prototype of some of the ideas presented in the discussion of log backup configurations. The purpose of the prototype implementation was to perform some benchmark runs and see how each interconnect algorithm affected the maximum average and burst speeds of the logging subsystem. The prototype was implemented using a beta version of INFORMIX-OnLine Dynamic Server™ version 5.0. Although we simulated the generation of log records, all other pieces of the prototype implementation were making use of the real OnLine Dynamic Server. This allowed us to isolate the logging subsystem of a real database system to benchmark the various interconnect algorithms. Due to hardware availability, we used a logging configuration made up of a single disk for the log device and a single tape for the backup device. We compare offline, synchronous, and asynchronous backup of the log device and examine their effect on performance.

6.1 The Implementation

INFORMIX-OnLine Dynamic Server uses a variation on ARIES’ [Mohan92] WAL protocol [Curtis88], and has several features which aided in the implementation. It uses the concept of "log files" and allows the database administrator to specify the number of log files as well as their size and location. A log file is a contiguous area of disk space that has been set aside to contain log data. The database administrator must
allocate a minimum of three log files. These log files can all be on one physical disk, or they can be placed on multiple disks. Allocating each log file to use a full disk makes a log file equivalent to what we have been calling a single log device.

OnLine Dynamic Server also has utilities which will perform an automatic backup of a log file to a specified backup device after it has become full and is no longer the current log file. This is equivalent to our offline backup if the log files are on separate disks. To implement offline backup of the log, we simply didn’t perform any backup of the log during the testing period. In a true implementation of offline backup we would have required a second log disk and waited to back up the first log disk until it was filled and we were using the second disk for the log. However, to measure the maximum burst speed using offline backup, it was sufficient to simply configure the system to avoid doing any log backup during the time of the benchmark. We also measured the log backup without any activity on the log disk to determine the maximum average speed when using offline backup (not taking tape switch times into account).

To implement asynchronous backup of the log device, we configured the system to have a large number of very small log files of 256K each, all on the same physical disk. As soon as the first log file was full, the log backup utility began reading the log data and writing it to the backup device. This allowed the log backup to stay within a minimum of 256K of the current point in the log. As the disk filled, the backup utility would compete for the log device and backup the log, thus implementing asynchronous backup of the log.

Implementing synchronous backup was more involved and required some code changes, along with the use of OnLine Dynamic Server’s mirroring capability. OnLine Dynamic Server implements mirroring in software and allows mirroring at the dbspace level, where a dbspace is a logical
grouping of disks or files. We put the log disk in a separate dbspace and modified the code to make the database server think that the log backup device was simply a mirror of the dbspace containing the log disk. We then modified the I/O subsystem to prevent seeks and reads in the log mirror to avoid any attempts at random access of the tape device. Version 6.0 of OnLine Dynamic Server also makes use of asynchronous I/O either through dedicated asynchronous I/O processes or direct calls to an OS kernel implementation of asynchronous I/O (on some platforms). Because of multiple processes capable of performing I/O, we also had to restrict the number of processes opening the backup device to one and send all writes to the backup (mirror) to this process, since multiple opens of a tape device were illegal. The asynchronous I/O capability allowed us to perform the writes to both the log device and log backup device in parallel. We waited for the status to be returned from both write operations before considering the write complete.

In order to generate the volume of log data we wanted on the given hardware, we instrumented OnLine Dynamic Server to be able to simulate the creation of some number of buffers of log data of a specified size. We modified the database server to allow a utility to send it a command to ask for the generation of the log data. This was done by creating a function in the OnLine Dynamic Server that would create dummy log records of 2K each (the OnLine Dynamic Server page size) until enough log records were generated to reach the specified write size, at which time we forced a flush of the records to disk. The placing of the log records into the log buffer and the flush of the buffer to disk were accomplished by making calls to OnLine Dynamic Server's log management subsystem. As indicated above, although the log records were simulated, all other pieces of the prototype implementation were making use of the real OnLine Dynamic Server.
6.2 The Benchmark

The main purpose of the logging subsystem benchmark was to determine the maximum burst speed when using a single log disk and single log backup device across each of the interconnect algorithms. We also used the benchmark data as well as some further tests to project what the maximum average speed would be for each log backup method.

For the logging subsystem benchmark, we varied the size of the log data written per log write across 4, 8, 16, 32, and 64K for each log backup configuration to determine its impact on performance. We varied the write size in order to examine the interaction between the size of log writes and the interconnect algorithm, recognizing that the performance of typical magnetic disks and magnetic tapes varies greatly at different write sizes. We stopped at 64K because that was the limit on the write size for the tape drive used in the benchmark (i.e., it couldn't handle writes of 128K). Each benchmark run consisted of writing 256M of log data of the particular log write size and measuring the performance of the log disk and log backup tape (where appropriate). We also performed similar benchmarks on the log disk and tape drive by themselves to determine their maximum speed for comparison with our logging speeds. To avoid extra flushes of the log buffer, we started each benchmark run with the current log file completely empty. This allowed each of the small log files to be filled exactly and prevented writes of partial log buffers due to reaching the end of a log file. (OnLine Dynamic Server will not allow a single log write to span log files.)

For asynchronous backup, all previous log data had been backed up prior to the start of each logging subsystem benchmark run so the backup was in sync with the current log position. For offline backup, no log backup took place during the benchmark run, so the speed of the offline backup indicated our maximum possible logging rate at each write size.
We also measured the log backup speed without any activity on the log disk to determine the maximum average speed when using offline backup, indicating the maximum speed at which we could backup log data. OnLine Dynamic Server prints a message into a designated message file up to once per second indicating the new current log file if it has changed. It also prints when a log file backup is starting and when it is complete. We were able to use these messages to determine how much data was being backed up during the course of the asynchronous backup benchmark runs, giving a good approximation of the maximum average speed when using asynchronous backup.

We used the same disk configuration for all logging subsystem benchmark runs regardless of interconnect algorithm to make sure no anomalies in performance occurred due to different log disk configurations.

The benchmark of the prototype implementation was performed on an 8 processor Siemen's RM600, running Sinix Version 5.41. The log disk was a 1.2G Hitachi DK516C drive, and the tape drive was an Exabyte 8500 tape drive. We created 4000 256K log files on the log disk, configuring 1G of the drive for log data.

6.3 The Results

This Section presents the results of our logging subsystem benchmark runs. To ensure that the numbers were all valid and repeatable, each benchmark was executed three times and the three runs were averaged to obtain a final result at each log write size. For each of the benchmarks, there was very little variation in the three runs used to obtain a final result.

Figure 6.1 shows the results from the offline backup and synchronous backup runs as well as the raw speeds of both the log disk and log backup tape. The offline backup numbers indicate the maximum speed at which we were able to generate and backup log records. Note that as the
log write size increased, the offline backup maximum burst speed fell behind the raw speed of the log disk. This behavior is an artifact of the configuration of the log disk. Because each log file was only 256K in size, we switched to a new log file very frequently during the course of the benchmark runs. The overhead of this log switch, in particular the messages that were printed in the OnLine Dynamic Server message log to indicate the new current log file, prevented us from fully saturating the log disk at the largest write sizes. In a separate benchmark using a log write size of 64K and a single large log file, we were able to fully saturate the log disk, equaling the raw speed of the disk.

As indicated above, we also measured the log backup speed without any activity on the log disk to determine the maximum average speed when using offline backup. We were surprised to discover that the average speed was actually about 20% slower than the maximum speed of the tape drive. This turned out to also be an artifact of the many small log files used in the test. We performed a separate benchmark backing up a single large log file and found that the backup was able to fully saturate the tape drive. Note that the average speed for offline backup is constant across all write sizes. This is because we backed up the log data using a 64K read and write size (optimum for both devices), which is not related in any way to the write size used to write the data in the first place.

The results for synchronous backup were as expected. The synchronous backup ran at the slower of the two devices for each run. At the 4K and 8K write sizes, the log disk was slower than the tape drive, and that was the speed at which the synchronous backup ran as well. At 16K, the log disk became faster than the tape drive, and the synchronous backup ran at the slower rate of the tape drive from there on out. These numbers demonstrate that both the maximum and average speeds of the synchronous interconnect algorithm will always be the minimum speed
between the log device and the log backup device. They also demonstrate the importance of being able to write large blocks of log data. At the smaller write sizes, we were actually saturating the devices in terms of the number of write operations they could handle per second rather than the bandwidth of the devices themselves.

![Figure 6.1: Offline and Synchronous Interconnect Results](image)

**Figure 6.1: Offline and Synchronous Interconnect Results**

Figure 6.2 shows the results from the use of the asynchronous interconnect algorithm. For comparison purposes, the figure also shows the speed of the offline backup method as maximums for the burst and average speeds. The severe slowdown experienced by asynchronous backup in maximum burst speed compared to offline backup is due to the contention for the log disk between the log record generator and the log backup process. We also discovered that the maximum average speed compared to offline backup is slightly reduced due to this contention.

At the 4 and 8K log write sizes, the raw tape speed is faster than the rate at which we can generate log data, meaning that the log backup should be able to stay very close to the current log write position.
For these write sizes, the average speed is the same as the burst speed. We expected the performance degradation in burst speed at these write sizes for asynchronous backup to be smaller than was actually the case. At 4K, we see a 16% degradation in maximum burst speed, and at 8K we see a 25% degradation. We believe that the amount of degradation was a little higher than expected because we were using 256K log files, which kept the log backup at least 256K behind the current log write position. Allowing the backup to get closer to the current log position, i.e. within one disk track, should reduce this degradation to closer to 10 to 15%. Note that this degradation necessarily forces the log backup to also operate at a 10 to 15% slower rate even though it is able to keep up with the log device. (We can't back up the log data any faster than it is written.) This reduces the maximum average speed by the same factor.

As the log write size increased, the difference between the rate of log record generation and log backup speed also increased. As the log record generation exceeded the speed of the tape drive from the 16 to 64K write sizes, the degradation in maximum burst speed ranges between 30 and 40 percent. This is due to the fact that the log write position is able to outdistance the log backup position by a larger margin,
increasing the seek time between the two. This added contention for the log disk also affected the maximum average speed and caused it to decrease by about 5%.

![Asynchronous Interconnect Results](image)

**Figure 6.2: Asynchronous Interconnect Results**

Figure 6.3 shows asynchronous backup burst performance at 25% intervals during the logging subsystem benchmark runs for each write size. After each interval of a benchmark run, we recorded the logging rate for that interval. This figure demonstrates the slowdown in logging rate after the logging subsystem reaches a steady state for the seek distance between the log record generator and the log backup. When the backup is able to stay close to the log record generator, the degradation is minimal over the course of the benchmark run. However, as the write size increases the degradation is more pronounced. Taken across a larger benchmark run than 256M, the degradation in burst speed for asynchronous backup may be even greater than the 30 to 40% measured in
our benchmarks. Writing more log data would lessen the impact of starting the log backup in sync with the current log position and increase the impact of the large seek distance between the two.

Figure 6.3: Asynchronous Interval Performance
In this thesis we have presented a model of the various pieces of a logging subsystem, including logging method, log device, log backup device, and log backup interconnect algorithm. As part of the model, we presented a criteria and weighting system as a method of evaluating a logging subsystem in order to determine the better of two configurations. We proceeded to examine three issues, namely log device bandwidth, log backup device bandwidth, and the log backup interconnect.

Our evaluation has shown that the use of striping and RAID provides the best solution for increasing the bandwidth of the log device at least cost. Along with our evaluation of log device bandwidth, we presented a new Append-Only RAID algorithm for optimal performance in a RAID log under varying workloads. Further, we have shown that the bandwidth of the log backup device must match the effective bandwidth of the log device and should be accomplished through the use of striping. Our evaluation of the log backup interconnect algorithm demonstrated that the use of inexpensive tape drives and the synchronous interconnect algorithm provides the best performance at least cost. Finally, we have demonstrated the performance impacts of each of the log backup interconnect algorithms using a prototype implementation in a commercial database system. Our prototype demonstrated the severe performance consequences of asynchronous backup, while demonstrating that synchronous interconnect will operate at the slower of the log device or log backup device.
Future work should include the effect of using alternative stable storage (such as battery backed up RAM or optical disks) for log devices and log backup devices. It should also include the effects of current trends in speed and cost characteristics for disks and tapes and their impact on the evaluation of the log backup interconnect. A further study should also include actual failure rates of tape drives and especially magnetic tapes to determine their impact on the conclusions of the thesis.
Bibliography


