Storage Management for E-Business

Presentation #331

by

Paul Massiglia

Melissa Stein

VERITAS Software

1600 Plymouth Street, Mountain View, CA 94043

650-335-8000

paul.massiglia@veritas.com

melissa.stein@veritas.com

Contents

Figures

Tables

Introduction: Storage Management for E-Business

The E-Business Phenomenon

Analysts agree that business on the Internet is growing exponentially. Forrester estimates that online business will grow from $43 billion in 1998 to $2.7 trillion in 2004. By the end of 1999, a sixth of U.S. households will be making online purchases. Ziff Davis Internet Trak predicts that by 2005 there will be 17 billion online purchase transactions annually. And it's not just a consumer phenomenon. Business-to-business electronic commerce accounts for 78% of total spending on electronic transactions in 1999.[1]

The reasons for going online are obvious. Being online makes a business more responsive, reduces costs, improves access to customers, shortens supply lines, improves cash flow, and helps manage inventory. If you aren't doing business online today, you'd better be planning to, because your competitors are.

Doing E-Business

Most people have been on the consumer side of an "amazon.com," "e*trade," or similar e-business experience. Today, most of the attention is focused on consumer purchases. But e-business is a lot more than that:

- Doing business online eliminates intermediaries and creates direct customer relationships.
- Electronic document interchange enables real time supply chain management. Orders can be placed, status verified, and invoices paid electronically, dramatically reducing turnaround time and errors.
- Intranets give employees instant access to up-to-date company information they need to function effectively.
- An electronic presence is a unique opportunity for a business to establish a strong brand with a wide following at very low cost.

Going online can streamline the way business is done today. For the future, it offers virtually limitless possibilities for new ways of doing business. It's a phenomenon that no organization can afford to ignore.

[1] Forecasts from quotes on http://www.sun.com/dot-com/whatis/index.html on October 11. 1999 and "IDC Predictions '99: The "Real" Internet Emerges," Frank Gens, Bulletin #W17971, December 1998

Information: The Lifeblood of E-Business

The essence of doing business online is connecting people with information. The information that makes e-business work is largely represented as electronic data stored and managed by computer systems. The partners that make up an e-business (employees, suppliers, and customers) need access to this information that is:

- ***reliable:*** The fastest way to destroy customer confidence is to be non-responsive. It doesn't matter if disks fail, applications halt, or equipment must be reconfigured, doing business electronically means information has to be available to respond to partner needs.
- ***fast:*** With e-business, competition is a click away. When customers or business partners ask for information, they expect instant answers, no matter how busy computer systems are.
- ***manageable:*** Information must be protected from equipment failures and human error. Online data has to be preserved, and moved to where the e-business needs it, often while it is being accessed.
- ***scalable:*** E-business growth opportunities are truly staggering. By one analyst's estimate a company can increase its revenues by as much as 50% just by going online. If an e-business can't grow its ability to deliver information as it grows, success can turn into failure.
- ***disaster proof:*** Going online can streamline operations, reduce costs and improve profitability. Once online, however, there's no turning back. Electronic data and applications have to be available, even if there's a fire, flood, or massive power outage. An e-business must be able to "keep on ticking" when the unforeseen happens.

This paper is about data storage management for e-business; on keeping electronic data available, providing fast access to it, managing it, enabling it to grow with the business, and making it disaster proof. It describes the use of storage management technologies to manage the storage that holds the information assets that enable e-business.

E-Business Data Requirements

Relational database technology is the obvious answer to getting an e-business online fast and efficiently. Database technology is mature and robust. Most of today's mission-critical applications are based on databases. Database management systems maintain transactional integrity,[2] even when multiple applications access data simultaneously.

Databases also protect data integrity. They store each data item once, no matter how many applications use it. Built-in filters prevent invalid values from entering the database. *Redo logs* help re-establish database consistency after system or application failure.

[2] Either the whole transaction is processed, or it is backed out and none of it is processed. Data is not corrupted by "half-done" transactions.

Database Management Systems for E-Business

E-business has some of the most challenging IT requirements in all of computing. On the one hand, 24x7 Internet business has to be at least as robust as the most mature applications in the corporate data center. On the other hand, the explosive growth in this young area means frequent reconfigurations as hardware is added or re-deployed, electronic files are moved and restructured, and applications evolve with business needs and practices. The question is how to provide customers, suppliers, and employees with reliable access to the information they need at performance levels that won't leave them frustrated, in the midst of all this chaos.

Database management systems clearly help bring order to the chaotic e-business environment. The fundamental concept of database management is to separate data from applications. Databases provide the stability, integrity guarantees, transactional semantics, and recovery capabilities that "keep data whole" as applications and IT infrastructure grow, change, and are replaced.

Online Storage for Database Management Systems

Database management systems are the obvious way to meet demanding e-business requirements. But they don't exist in a vacuum. The higher the quality of its underlying storage, the better the job a database management system can do. Database management systems require three attributes from storage:

➡ ***Reliability***. Database management systems organize the contents of huge numbers of disk blocks into interrelated tables of user data and metadata that collectively represent the state of an e-business. They are great at maintaining data's logical integrity, but they need the support of a solid foundation to maintain data's *physical* integrity.

➡ ***High performance***. To deliver the integrity guarantees they promise, database management systems carefully sequence I/O operations, including significant numbers of overhead operations that enforce integrity and enable recovery after system failures. While database management systems attempt to minimize physical I/O through extensive use of cache, disks ultimately have to be read and written. Higher performance data access means greater application responsiveness.

➡ ***Ability to grow non-disruptively:*** Database management systems generally handle growth very well. Most support the addition of storage capacity to online databases. In order to exploit these capabilities, however, storage structures must be able to grow dynamically as well. Not only must it be possible to add storage to an online database; the I/O workload must be dynamically re-balanced across all storage resources to avoid *hot spots* that over-utilize some disks and buses while others are idle.

E-Business Storage Choices: Disks or Volumes?

Database administrators must decide whether to use "raw" disk partitions or logical *volumes* as containers for database tablespaces and metadata. Raw partitions are ranges of consecutively numbered disk blocks that are addressed by I/O requests made directly to I/O drivers. Volumes are logical combinations of disk blocks presented by *volume managers* implemented in host software like the *VERITAS Volume Manager* or (in more limited form) in RAID controllers. Volume manager's role is to subdivide or combine physical disks to achieve desirable performance and availability properties. Volume managers can:

- partition a large disk into smaller volumes,
- concatenate or stripe part or all of the storage capacity of several disks to appear as one larger volume,
- *mirror* identical copies of data on two or more disks or sets of disks, and,
- use *parity RAID* to protect against data loss due to disk failure.

Volume managers exist because applications need online storage that is bigger, faster, or more reliable than the biggest, fastest, most reliable disks available on the market. In general, database users favor managed volumes with some minor reservations:

- ***Data integrity.*** Some volume managers use write cache to enhance performance. Without appropriate safeguards, this can make it difficult for the database management system to protect data integrity in the event of system crashes.
- ***Integration.*** Some volume managers, particularly those implemented in RAID controllers, are not closely integrated with host operating environments. For the most part, their volumes emulate SCSI or Fibre Channel disks. It can be difficult to exploit volume features that don't have disk counterparts, such as dynamic capacity expansion, or non-disruptive failures (e.g., a single disk failure in a parity RAID volume).

Despite these perceived shortcomings, neither of which is characteristic of the VERITAS Volume Manager, volumes are generally preferred as a storage foundation for database management systems.

E-Business Storage Choices: File Systems or Partitions?

Intuitively, it seems that files should make ideal "containers" for database tables and indexes. Files are convenient and manageable. They can expand when required. They can be moved from place to place with simple, easily understood management operations. They are natural objects for backup.

Counterbalancing these advantages, conventional file systems also have some perceived disadvantages when used as database storage:

- ***Weak persistence guarantees***. Most file systems compromise between application performance and data integrity by using write-back cache. Application write request completion is signaled before data is actually written to disk. Writes are done *lazily*, possibly long after applications have acted on the assumption that data was safely stored. While this improves application performance, system crashes with unwritten data in cache can cause updates to be irretrievably lost, giving rise to "debit without matching credit" database scenarios.

- ***Undesirable performance characteristics***. Most file systems move data to a private buffer cache before writing it to disk. This ensures that the correct data is written, even if a poorly designed application inadvertently overwrites its memory before the write is complete. Moving large amounts of data from one part of memory to another exacts a heavy performance penalty. Database management systems have no need for this protection. For them, I/O directly from their own memory space is preferable.
- ***Sub-optimal sharing semantics***. Most file systems allow any number of applications to read from a file simultaneously, but allow only one to write it. For most applications, this makes sense. Two applications writing a file simultaneously can destructively interfere with each other's updates. Most database management systems consist of many concurrent execution threads, so they look like many separate applications from a file system standpoint. Unlike typical applications, however, the threads of a database management system are highly coordinated. Two or more threads only access a data object simultaneously when they can do so without causing inconsistency. Restricting access to one thread at a time constrains database performance unnecessarily.

For these reasons, conventional file systems are sometimes thought to be inappropriate storage containers for databases. However, new file system technologies can overcome these disadvantages, making them a clear choice for e-business database storage.

Total Data Storage Management for E-Business

For a database to perform optimally under all of the unpredictable circumstances that are a part of doing business online, it needs an underlying infrastructure that:

- provides a robust, high-performance storage environment,
- enables backup and other management operations during database operation,
- supports database growth over a wide range of capacities, and,
- maximizes data availability, even if disaster strikes.

Collectively, VERITAS technologies provide such an infrastructure for e-business databases. VERITAS calls this infrastructure *Total Data Storage Management*. *Total Data Storage Management* for e-business databases is an integration of multiple software technologies that span the I/O and storage management spectrum.

The Data Access and Management Stack

To help position the components of *Total Data Storage Management*, Figure 1 depicts a data access and management view of an application execution environment. Borrowing an analogy from networking, Figure 1 can be thought of as a *data access and management stack* for e-business applications.

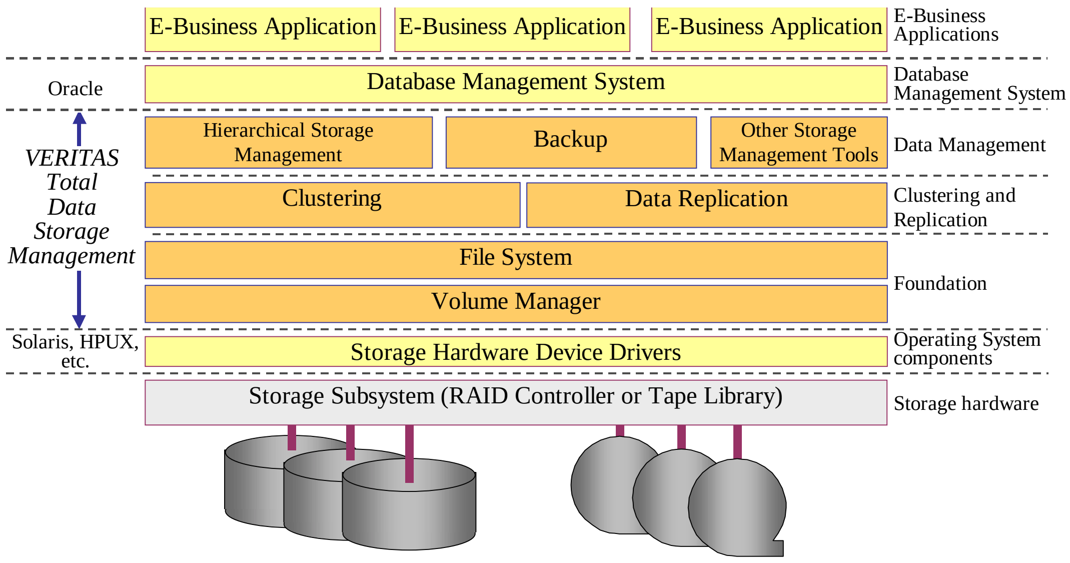

E-Business Application
E-Business Application
E-Business Application
E-Business Applications
Oracle
Database Management System
Database Management System
VERITAS Total Data Storage Management
Hierarchical Storage Management
Backup
Other Storage Management Tools
Data Management
Clustering
Data Replication
Clustering and Replication
File System
Volume Manager
Foundation
Solaris, HPUX, etc.
Storage Hardware Device Drivers
Operating System components
Storage Subsystem (RAID Controller or Tape Library)
Storage hardware

Figure 1: An E-Business Data Access and Management "Stack"

As Figure 1 illustrates, e-business database applications require a substantial software infrastructure. VERITAS *Total Data Storage Management* provides this infrastructure. This paper organizes the discussion of *Total Data Storage Management* software technologies into:

- ***Foundation technologies***. File systems and volume managers can be used together or individually to create a high-performance, flexible, scalable e-business database foundation. They enable data striping, RAID, point-in-time snapshots, block-level incremental backup, online storage expansion and reconfiguration, and rapid recovery from system crashes.

- ***Data replication technologies***. Replication enables timely distribution of information to geographically distributed data centers, where it can be used to shorten communication lines or to enable recovery from data center or environmental disasters.

- ***Clustering technology***. Clustering supports the "failover" of applications to alternate computers when the computers on which they are running experience problems. For some applications, clustering also enables incremental growth through the addition of new computers to existing clusters.

- ***Data management technology***. E-business information must be *managed* in an environment of 24x7 online operations. The explosive growth that typically characterizes e-business makes the problem even more challenging there than in conventional IT environments. Database management system aware backup ensures transactionally consistent database backups, and also maintains up-to-date copies of program images and other ancillary files for fast restores of the operating environment if necessary.

The next four sections of this paper examine these technology groupings in detail and illustrate how they combine to meet e-business database storage management needs.

I. The Foundation

24x7 High Performance Online Access to Information

The Nature of E-Business Database I/O

Integration of volume management and file system technologies offers a highly available, high performance, scalable storage foundation for e-business databases. To appreciate why *Total Data Storage Management* is ideal for e-business databases, it is helpful to consider the nature of e-business database I/O.

E-business applications are typically *I/O request intensive*. They make large numbers of I/O requests per second, but each request only transfers a few kilobytes of data. The order in which application requests occur is unpredictable, so an e-business database management system accesses its storage randomly. The combination of random access and small I/O requests means that *accessing* data (getting a disk read/write head to the right spot on the right disk platter) is more important to e-business database I/O performance than *transferring* data.

Volumes: Data Integrity and I/O Performance

The first requirement of a 24x7 application is that its storage be available all the time. Disks, I/O buses, and even computers can fail, but for e-business success, information must be continuously accessible. Volume managers aggregate physical disks into virtual *volumes* that can survive disk and I/O bus failures. From the standpoint of database management systems and other applications, volumes are functionally identical to physical disks. They can survive disk failures, however, through the use of *host-based RAID* technology. Roughly equivalent to disk subsystem RAID, host-based RAID volumes offer two key advantages:

- ➨ ***Initial and incremental hardware investments are significantly lower***. When an e-business first begins to operate, it typically doesn't have much online information. Because host-based RAID can work with low-cost commodity disks attached to low-cost commodity host bus adapters, it can protect a small amount of data without a large storage hardware investment. As the e-business grows and has more online data, host-based RAID can enable growth in increments as small as a single disk, unlike hardware RAID subsystems, which often require larger incremental investments.
- ➨ ***Host-based volumes complement RAID disk subsystems***. As an e-business grows and has more applications accessing more online data, hardware RAID may eventually be required, even if

only for physical packaging reasons. Host-based volumes can aggregate hardware RAID arrays for increased capacity or improved availability (for example by mirroring data across disk subsystems).

What Kind of RAID?

Table 1 lists the availability characteristics of the various volume-supported types.

Volume Type	Data Protection (Availability)	E-Business Database Application
Mirrored (2x or more) (sometimes called RAID-1)	Very high	Smaller (single-disk) critical data objects
Striped and Mirrored (two or more copies)	Very high	Large (multi-disk) critical data objects
Striped with Parity ("RAID-5")	High	Important "read-mostly" data
Striped without Parity (sometimes called RAID-0)	Lower than single disk	Minimal applicability; (easily replaced low-impact data)
Concatenated (capacity aggregated)	Lower than single disk	Minimal applicability; (easily replaced low-impact data)

Table 1: E-Business Applications of RAID: Availability

As Table 1 indicates, mirrored volumes provide the best protection against disk failures, simply because with more disks holding redundant data, they can survive more (but not all) multiple failure modes. Moreover, when a disk does fail, the impact of the failure on application performance is lower than with parity RAID. The volume manager is capable of mirroring data across portions of two or more disks, across two or more entire disks, or across two or more multi-disk sets, as Figure 2 illustrates.

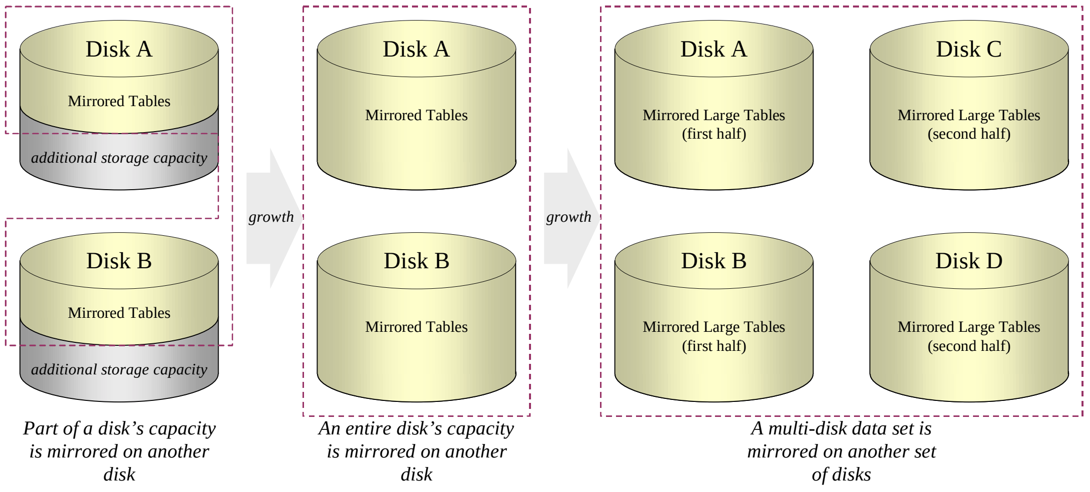

Disk A
Mirrored Tables
additional storage capacity
Disk B
Mirrored Tables
additional storage capacity
Part of a disk's capacity is mirrored on another disk
growth
Disk A
Mirrored Tables
Disk B
Mirrored Tables
An entire disk's capacity is mirrored on another disk
growth
Disk A
Mirrored Large Tables (first half)
Disk C
Mirrored Large Tables (second half)
Disk B
Mirrored Large Tables (first half)
Disk D
Mirrored Large Tables (second half)
A multi-disk data set is mirrored on another set of disks

Figure 2: VERITAS Volume Manager Mirroring Configurations

This flexibility enables an e-business to start up at modest cost, and add storage capacity as it grows, without major investment increments. Online management utilities support the movement of data between volumes (as suggested by the “growth” arrows in Figure 2) while the data is in use by applications.

Data Striping to Balance I/O Load

Volume managers also support the *striping* of data across several disks to balance load and thereby optimize performance. Data can be striped across disks in three configurations:

- ***With no failure protection***. This configuration, sometimes called *RAID-0*, balances I/O load across available disk resources, but affords no protection against disk failure. It should be used sparingly for critical e-business data.
- ***With parity protection***. This configuration is often called *RAID-5*. It offers lower cost failure protection than mirroring, because one “check data” disk protects an arbitrary number of user data disks against the failure of any one of them. RAID-5 has a write performance penalty, however, that makes it unsuitable for frequently updated tables. In e-business applications, RAID-5 is suitable for data that is seldom updated, such as static web pages or data warehouses.
- ***With mirroring***. This configuration is sometimes called *RAID-1+0*. It adds the failure tolerance of mirroring to the load balancing provided by striping. It is the preferred combination of failure tolerance, performance, and practicality for business critical online data. Striping with mirroring should be used for all mission critical e-business information, such as sales and financial records, as well as for any frequently updated data, such as inventory, manufacturing, or shipping records.

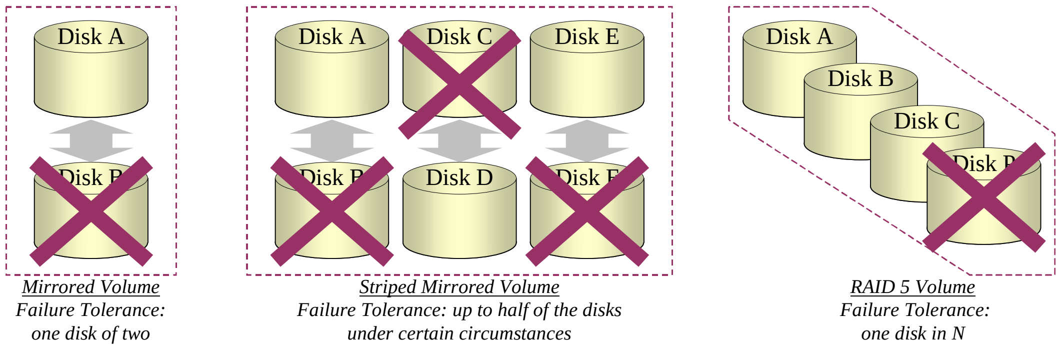


Figure 3: Failure Tolerance of Different Types of RAID Volumes

Using Volume Technology to Meet E-Business Performance Needs

In addition to providing highly available storage containers for online data, managed volumes generally outperform raw disks. Each type of RAID volume affects I/O performance slightly differently. When designing an e-business database, an awareness of RAID volume characteristics is helpful for optimal data placement. Table 2 summarizes RAID volume performance characteristics.

RAID Volume	Performance	E-Business Database Application
Mirrored (2x or more) (sometimes called RAID 1)	High read performance Average write performance	Single-disk critical data objects e.g., HR database
Striped and Mirrored (two or more copies)	Very high read performance High write performance	Large multi-disk critical data objects e.g., operational sales or mfg. data
Striped with Parity ("RAID 5")	High read performance Low write performance	Read-mostly data e.g., web pages, data warehouse
Striped without Parity (sometimes called RAID 0)	High read and write performance	Minimal applicability; (easily replaced low-impact data)
Concatenated (capacity aggregated)	Average read and write performance	Minimal applicability; (easily replaced low-impact data)

Table 2: E-Business Applications of RAID: Performance

As Table 2 indicates, mirrored volumes typically outperform single disks when reading. This is because the volume manager can minimize I/O latency by choosing the least busy of a mirrored volume's disks to satisfy each read request.

Expected write performance with two-way mirrored volumes is comparable to that of a single disk. The volume manager must mirror each application write request on each of the volume's disks, however the writes are independent of each other (unlike RAID-5 writes), and can be done concurrently. RAID-5 writes tend to perform poorly relative to single disk writes because each application write requires that the volume manager perform a series of reads and writes to the RAID volume's disks, as well as an intent log entry. For this reason, RAID-5 volumes are not recommended for frequently updated e-business database tables.

A File System for E-Business Databases

Volume managers provide a robust, high performing foundation for e-business databases. Specialized file systems add high-performance, flexible, manageable containers suitable for database tables and indexes.

As enumerated earlier, the primary limitations of file systems for use as database containers are:

- ***Weak persistence guarantees or poor performance***. Most file systems signal I/O completion to applications before data is actually on disk. This makes it impossible for database management systems to guarantee data persistence (crash recoverability), and therefore data integrity. Alternatively, a database management can force the file system to flush its cache after critical writes, resulting in poor multi-stream performance.
- ***Undesirable performance characteristics***. Most file systems move data to kernel buffer cache before writing it to disk. This creates high processing overhead that is unnecessary for databases.
- ***Inappropriate sharing semantics***. Most file systems require that an application wishing to write a file set a kernel lock. For database management systems, this is unnecessary overhead that also reduces parallelism.

Enhanced file system technologies, like those from VERITAS have none of these limitations. Moreover, additional advantages for files used as containers for e-business database tables are available. Many of the advantages stem from the file system's *extent-based* architecture.

Conventional and Extent-Based File System Data Allocation

In both conventional and extent-based file systems, the on-disk data structures that define files (inodes) contain storage space descriptors that indicate which blocks of the underlying disk or volume are occupied by the file's data area. Each storage space descriptor contains a starting volume block number and indicates or implies a number of blocks. Since inodes have a limited amount of space for storage space descriptors, files that require many descriptors must have auxiliary structures linked to their inodes. Reading and writing such files generally requires "extra" disk I/O to retrieve these descriptors before file data can be accessed.

An extent-based file system allocates storage space in variable length *extents*, rather than in the fixed-length allocation units of one to eight kilobytes used by conventional UNIX file systems. Each extent is an arbitrarily large set of consecutively numbered disk blocks, and can thus be concisely described by a starting block number and a block count. Because each extent can be arbitrarily large, a single inode often contains the entire block map, even for very large files, resulting in low overhead file access.

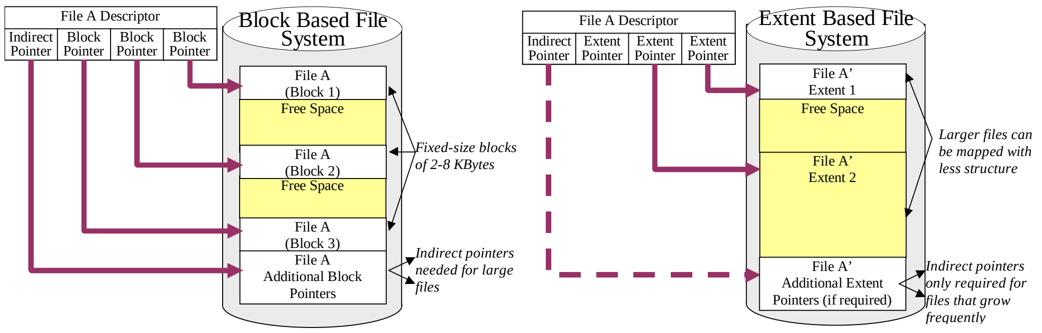


Figure 4: Extent Based File System

For database management systems, the key advantage of extents is that they enable larger contiguous storage areas to be mapped with minimal overhead. Large contiguous storage areas offer several benefits for database management systems:

- ***More compact storage descriptor structures***. Because each descriptor can map more blocks, fewer descriptors are required to map a file. Fewer descriptors means faster translation between the file offsets specified in application requests and the disk or volume addresses required in I/O requests.

- ***More efficient table scans***. Larger contiguous storage areas enable the use of larger I/O requests for sequential table scans. This reduces both I/O processing overhead and time lost waiting for disks to rotate to the starting point for the next sequential data transfer.
- ***Reduced disk fragmentation***. Larger contiguous storage areas reduce *disk fragmentation*, a condition in which logically contiguous data are stored in multiple non-contiguous fragments of disk capacity. Disk fragmentation reduces I/O efficiency because some application requests must be split into multiple disk requests for non-contiguous blocks of data.
- ***Aggressive sequential I/O policies***. Contiguously located data can be read ahead during sequential table scans. This decreases access latency. Moreover, some RAID controllers attempt to consolidate contiguous application write requests, for more efficient disk utilization. This is sometimes called *I/O request clustering*.
- ***Larger files***. Extent-based storage allocation makes large files practical. The VERITAS file system supports individual files of up to two terabytes in size in operating system environments that can support them.

Pre-Allocation for Efficiency and Application Robustness

In non-database applications, file creation, extension and deletion are relatively frequent. Databases, however, have relatively static table structures. Creation, extension and deletion of database datafiles are infrequent (although they can and do happen).

File systems allow storage space to be reserved when files are created (before they are written). Pre-allocation has two advantages for e-business databases:

- ***Optimal data layout.*** The database administrator building a new database can use space reservations to establish an optimal datafile layout for known requirements.
- ***Minimization of database space allocation failures.*** If database datafiles are pre-allocated, transactions and batch jobs that create temporary files will not cause database space allocation failures.

Enabling communication between Oracle and the file system for the purpose of aligning pre-allocated datafiles with file system extent boundaries. This minimizes “split” database pages that cross extent boundaries and allows the database management system to take maximum advantage of I/O request clustering.

The combination of extent-based architecture and pre-allocation of storage space are the basis for another VERITAS technology feature that makes file systems perform like raw partitions—*Quick I/O for Databases*.

Quick I/O for Databases: Data Persistence and Low Overhead

Quick I/O for Databases, part of the VERITAS Database Edition *for Oracle* integrated suite, enables a database management system to do I/O to pre-allocated files as if they were UNIX raw devices. A

UNIX symbolic link to each file provides the database management system with raw device access. Normal file system access paths are also available. Four factors make *Quick I/O* access performance equivalent to that of raw disk I/O:

- ***Direct I/O.*** With *Quick I/O*, data moves directly from database management system memory (e.g., Oracle's Shared Global Area) to disk, with no intermediate copy to kernel space. This eliminates substantial processing overhead.
- ***Elimination of redundant caching.*** Since database management systems have their own cache, operating system cache is redundant. *Quick I/O's* direct I/O eliminates this redundancy.
- ***Enable more asynchronous I/O.*** Using raw I/O, a database management system can make many concurrent I/O requests. This increases parallelism, and also gives drivers, controllers, and storage devices larger I/O request lists to use in optimizing device performance.
- ***Minimize kernel file locking.*** Kernel file locks are bypassed. Since database management systems do their own internal locking, kernel file locking is redundant and leads to excessive serialization. Eliminating kernelized locking maximizes database management system parallelism, and loses nothing in data integrity.

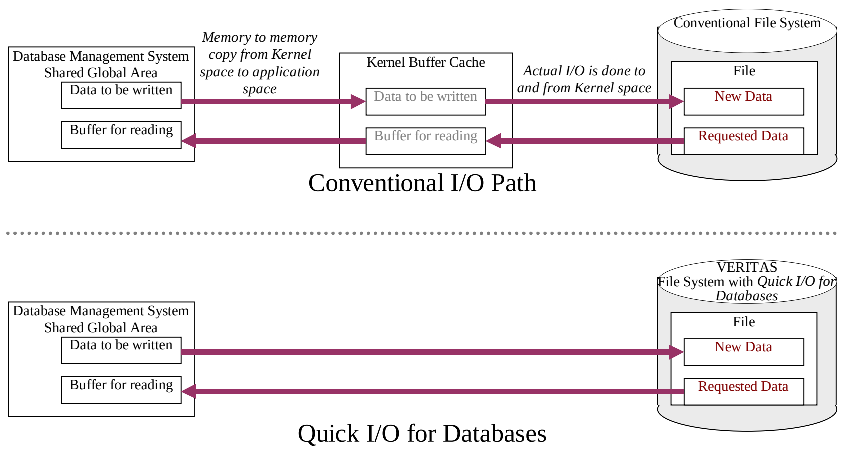


Figure 5: Quick I/O for Databases Supports Direct I/O from Application Space

Using *Quick I/O* sacrifices no file system functionality. Files used with *Quick I/O* can also be accessed through normal file system paths, for example for backup, replication, or other management purposes.

Fast Recovery

Even in the best-prepared e-business installation, things can go wrong. Applications, hardware components, or operational procedures can all cause system crashes. A recovery plan for getting back online quickly is vital. An important part of recovery is getting applications restarted as soon as possible.

Special techniques for recovering Oracle databases have been discussed earlier. Equally important, however, is recovery of files that aren't in the database—the dozens or hundreds of configuration, log, and other ancillary files that are needed to make applications run.

When a busy system crashes, the integrity of its file system structures is in doubt. Files may have been created, extended, or deleted, but persistent (on-disk) file system metadata may not yet be completely updated. A crash at the wrong time can lead to lost files or storage space, or worse, blocks allocated to two or more files.

The usual means of verifying file system metadata integrity is to run a program to check and repair structural integrity before the file system is mounted and used. In UNIX systems, this program is commonly called *fsck* (file system check). The fsck program and others like it validate the structure of a file system, and make sure that no disk blocks are lost or multiply allocated. They may undo partial metadata changes, losing recent updates, but the file system is left structurally intact. The problem with using *fsck* in e-business is that with large file systems it can take a long time to run (2-5 minutes per gigabyte of data by one estimate). Since no applications, including the database management system, can begin executing until *fsck* is finished, recovery time can be very long, reducing system availability.

VERITAS file system technology incorporates an alternate first-level recovery technique called *intent logging*. Each time the file system's structure is to be changed (e.g., when a file or directory is created, extended, or deleted), all of the file system metadata updates required to effect the change are logged in a file system private area before any are performed. The file system intent log is a small circular buffer, designed to be self-clearing. Structure changes that would result in overwriting intent log entries whose execution is not complete are stalled until log space is available.

When recovering after a system failure, the file system reads the intent log, and makes sure that all metadata updates in it have been applied completely. Since the number of incomplete metadata updates at any instant is typically very small compared to the total amount of metadata in a large file system, reprocessing the intent log is normally much (orders of magnitude) faster than running a full *fsck*. Intent logging allows file systems to be mounted sooner, enabling the database management system to start database recovery earlier, ultimately increasing application availability.

Online Administration

Because e-business is in its infancy, organizations' e-business information systems will grow and change as time passes. Growth and change both imply that large amounts of data must occasionally be moved from one location to another:

- databases outgrow their storage capacity, which must be expanded or replaced,
- databases must be reorganized within their volumes to improve performance, and
- databases must be moved from one computer or geographic location to another.

The need to reorganize, expand, and relocate data is counter to the fundamental requirement of e-business—maximizing availability. Online file system administration allows the most frequent administrative functions to be performed while files are in use.

Online Defragmentation

As applications run, files are created, extended, truncated and deleted. Over time, volumes become *fragmented*. On a fragmented volume, as the right side of Figure 6 illustrates, files and free space are intermixed. The free space available for new files may be in many small ranges of contiguous blocks. A file creation request for a large contiguous extent of space may fail, even though there is adequate space available on the volume. Non-contiguous allocation requests will succeed, but the high overhead of accessing a fragmented file can lead to poor performance.

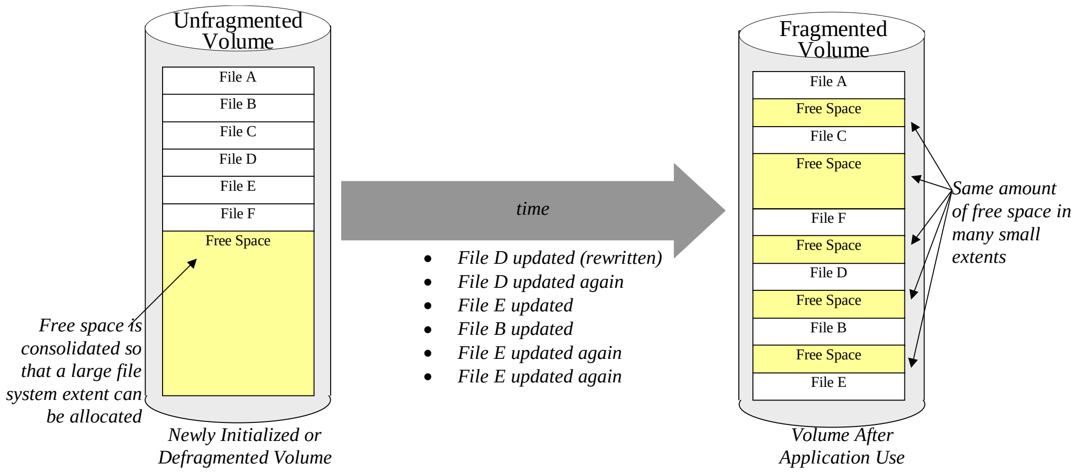


Figure 6: Volume Fragmentation

Online file system *defragmentation* can remedy this. The file system can move a file from one set of blocks to another and update its metadata to reflect the file's new location. This is transparent to applications, which access the file identically before, during and after the move. If all the files on a volume are made contiguous, the free space on the volume is consolidated, and the volume is said to be *defragmented*, as the left side of Figure 6 illustrates.

Moving a file that is not in use is easy. It is more of a challenge if the file is being accessed by applications while it is being moved.

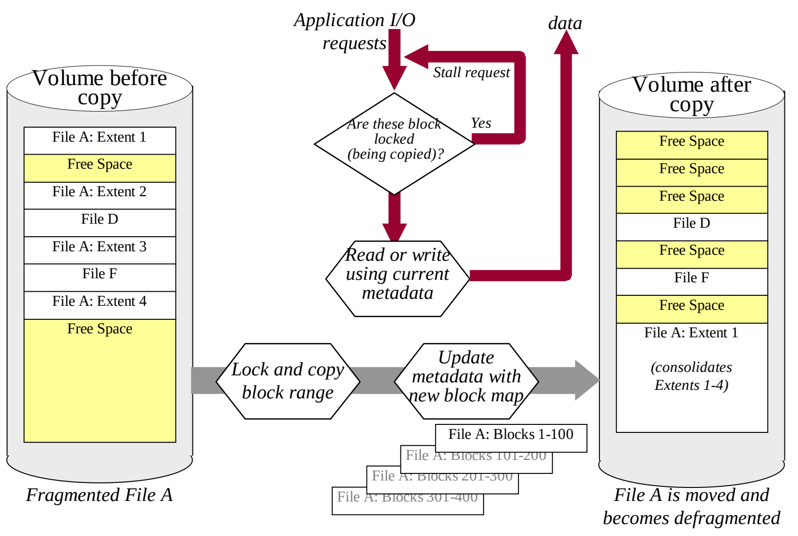


Figure 7: Movement of an Open File

Figure 7 illustrates how files are moved while they are being used by applications. To move a file, the file system first allocates destination space for the entire file. Ranges of blocks in the file are then locked and moved. File system metadata is updated to reflect the new volume addresses for the range of blocks just moved, and the block range is unlocked to allow application access. If an application attempts to access blocks that are being moved, its I/O request is stalled until the blocks are again available. If the file system attempts to move a range of blocks that are locked by applications, its I/O stalls until the application releases the blocks.

Moving Database Container Files

Database administrators move tablespaces while they are in use to satisfy the needs to extend, relocate, or reconfigure tables that inevitably arise in rapidly growing e-business environments. For example:

- Storage policy changes, such as a change from simple or RAID- 5 volumes to mirrored volumes for tables whose importance or update frequency has increased, can be implemented while the tables are in use.
- Tables that grow dangerously close to the capacity of their devices can be moved to larger devices or devices with more available capacity.
- Files can be moved from failing storage devices that are identified, for example, by RAID controller predictive failure analysis capabilities.

The ability to move files while they are in use enables these and many other management capabilities that can mean the difference between success and failure for an e-business computing environment.

Online Expansion

Perhaps the most frequently encountered reason to move files while they are in use in e-business is online capacity expansion. As an e-business matures:

- its set of products and services may increase,
- it gains more customers,
- its transaction history grows, and,
- the number of its online applications increases.

All of these factors require that more information be kept online, which in turn means storage growth. Oracle tablespaces can be expanded while they are online, either automatically or by database administrator action, provided that their containers are expandable. Raw partitions and some ordinary file systems permit expansion, up to partition or file system capacity. With foundation technology, online volumes can be expanded by the addition of storage capacity. File systems can then be expanded to use the new capacity, also while online.

Point-in-Time Database Images: Snapshots

E-business constant availability requirements conflict with the good general business practice of regular backup. A backup copy of a database is of little value unless it represents the state of the business at a single point-in-time. Backup copies should therefore be made when no database transactions are in progress. Since an online database can be updated at any time, it is difficult to guarantee that backing up a database while it is online will result in a consistent point-in-time image of data.

One way to achieve point-in-time backup is to stop using the database for the duration of the backup. Backups made in this way are often called *cold* database backups. Since there is no database activity during a cold database backup, it is the fastest full database backup technique. To perform a cold database backup, a system administrator must find a so-called *backup window* during which it is acceptable for the database to be unavailable. Traditionally, backup windows have been during late night hours or weekends. With e-business, however, data must be just as available during these times as during the business day. Cold database backup therefore does not meet e-business availability requirements. Moreover, as a business grows, the amount of its online data increases. Backup time grows proportionally, but the backup window remains the same—cold database backup becomes a less and less viable option.

The ability to create a point-in-time database image or *snapshot* resolves this difficulty. Snapshots are point-in-time images of the state of a file system created and maintained by the file system itself. Snapshots are transparent to database management systems, except at the moment of creation. A file system for which a snapshot is created is called a *snapped* file system. It can be updated as usual by applications after snapshot initiation. The point-in-time image, which appears to applications as a read-only file system, is called a *snapshot*.

Snapshot creation takes only seconds. The file system uses unallocated space to create a *modified block map* in which it will record changes to the snapped file system. It also allocates some space for a *changed block area* in which it will record the prior contents of any block changed after snapshot initiation. Once these structures have been created, both snapped and snapshot file systems are available to applications.

How Snapshots Work

Snapshots use a *copy-on-write* technique to maintain point-in-time file system images. The first time after snapshot creation that an application writes a given block to a snapped file system, the file system copies the block's contents to the snapshot's changed block area. It then updates the snapshot's modified block map to indicate that the before image of a modified block is stored in the snapshot area. Finally, it overwrites the block in the snapped file system. The storage space consumed by a snapshot is proportional to the amount of data updated during the time that the snapshot is active.

When an application, such as backup, reads a block from a snapshot, the file system first checks the modified block map to determine whether the block has been modified since the instant of snapshot creation. If the block is unmodified, it is read from its original location in the snapped file system. If the block has been written since the instant of snapshot creation, the modified block map indicates the location of its original contents in the snapshot's changed block area, and it is read from there.

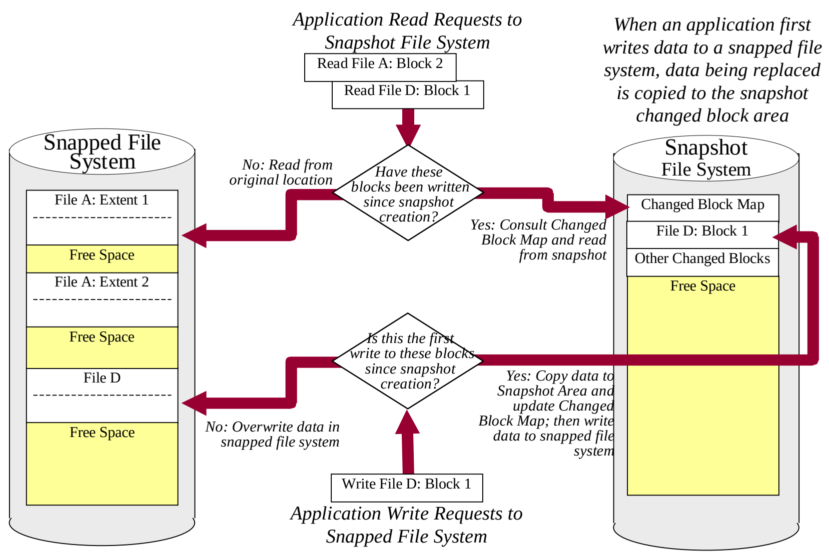

Application Read Requests to Snapshot File System
Read File A: Block 2
Read File D: Block 1
When an application first writes data to a snapped file system, data being replaced is copied to the snapshot changed block area
Snapped File System
File A: Extent 1
Free Space
File A: Extent 2
Free Space
File D
Free Space
No: Read from original location
Have these blocks been written since snapshot creation?
Yes: Consult Changed Block Map and read from snapshot
Snapshot File System
Changed Block Map
File D: Block 1
Other Changed Blocks
Free Space
Is this the first write to these blocks since snapshot creation?
Yes: Copy data to Snapshot Area and update Changed Block Map; then write data to snapped file system
No: Overwrite data in snapped file system
Write File D: Block 1
Application Write Requests to Snapped File System

Figure 8: Operation of a File System Snapshot

Snapshot Consistency

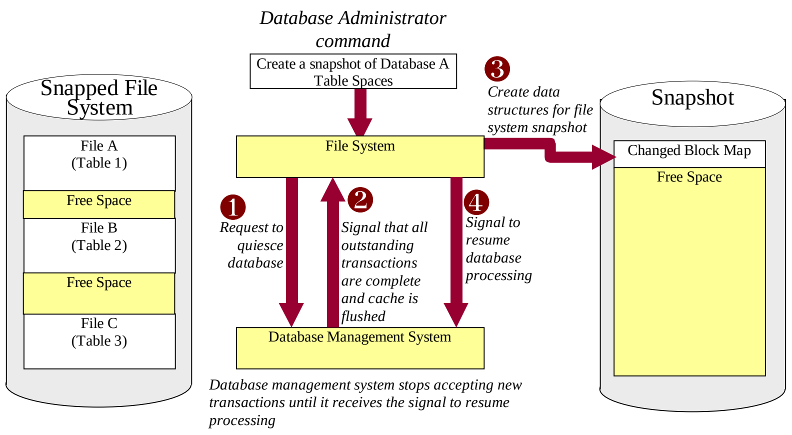
For a snapshot to be a useful source for backup, it must not only represent a point in time image of a file system, but also a database quiet state in which:

- no partially complete transactions are in progress, and,
- all completed transactions are reflected on disk (not just in cache.)

A database in this state is said to be *quiescent*.

When an administrator initiates a snapshot, a file system can stall I/O requests and flush (write to disk) any cached user data and metadata to make its disk image consistent. However, a file system cannot ascertain whether a database management system using it for storage has transactions in progress. A database might have transactions in progress with no I/O requests outstanding. [3]

A file system administration tool can request Oracle to complete all outstanding transactions, flush its cache, and stall any new transactions so that a consistent snapshot can be initiated. Once a database is quiescent, the file system initiates a snapshot, an operation that takes a second or two. Once the snapshot has been initiated, the database is released to resume transaction execution. This minimizes the interval during which the database is unavailable to applications. Figure 9 illustrates snapshot creation.



Database Administrator command
Create a snapshot of Database A Table Spaces
Snapped File System
File A (Table 1)
Free Space
File B (Table 2)
Free Space
File C (Table 3)
File System
1 Request to quiesce database
2 Signal that all outstanding transactions are complete and cache is flushed
3 Create data structures for file system snapshot
4 Signal to resume database processing
Database Management System
Database management system stops accepting new transactions until it receives the signal to resume processing
Snapshot
Changed Block Map
Free Space

Figure 9: Creating a Snapshot of a Database

Once created, a snapshot behaves like a read-only file system. Database, file, or raw device backup programs can read it as a source for making backup copies. Data read from the snapshot is always identical to what would have been read from the file system at the instant when the snapshot was created. The contents of a backup made from a snapshot are thus exactly equivalent to those of a cold database backup. The difference is that a database may be in use by applications while its snapshot is being backed up.

Snapshot techniques, online defragmentation, Quick I/O, and extent-based file system allocation all help to increase the performance, manageability, and availability need of e-business environments. From the core foundation technologies we move to replication to ensure data is accessible where it is needed.

[3] For example, transactions involving humans often incur “think time,” during execution. No database I/O is outstanding during these periods because the human is pondering the next action.

II. Replication

The Right Data in the Right Place at the Right Time

Why Replication?

In a typical e-business, it doesn't take long for growth to create a need for multiple processing sites. The responsiveness requirements of customers, suppliers, and employees usually make it most efficient to locate information close to where it is used. If information is to be made available from multiple sites, data must be *replicated*, or distributed, to those sites. Price lists, product specifications, web pages, and so forth are often replicated at all of an e-business' operating locations. Obviously it is important that this type of data be identical everywhere in the organization. Changes must therefore be *synchronized* across all locations.

Data is also frequently replicated for *data mining*. As an e-business matures, it acquires an ever-growing body of historical data about its operations and its customers. Businesses have discovered that historical data can be stored in *data warehouses* and *mined*, or analyzed for trends that can be used for a variety of planning purposes. While it is extraordinarily useful, data mining is I/O intensive. An e-business' online data usually cannot be mined without severely impacting operations. To avoid adverse operational impact, data can be replicated to a separate server (the data warehouse or data mart), where it is mined while business operations continue.

The third, and perhaps most important, reason for replicating data is *disaster recovery*. As an e-business grows, the economic and social consequences of a data center-disabling event can become significant. Prudent business practice dictates that for its own survival, an e-business should be able to recover quickly from a fire, flood, vandalism, power grid failure, software failure, or other event that could completely incapacitate a data center.

The Nature of Replication

Whether the purpose is data distribution, mining, or disaster recovery, the nature of data replication is the same. Up-to-date copies of operational data are kept online at one or more sites separate from the main data center, and these copies must remain in synchronization with the primary databases used to operate the business.

Storage Replication Technologies

Replication keeps one or more secondary copies of e-business data in synchronization with a master or primary copy that is being processed by applications. VERITAS replication technologies can replicate either:

- from a *primary file system* to one or more secondary file systems on different computers, or,
- from a *primary volume group* to one or more secondary volume groups on different computers.

Both file and volume group replicators are designed to maintain the integrity of replicated data in the event of network or system outages, while at the same time providing adequate performance for operational practicality.

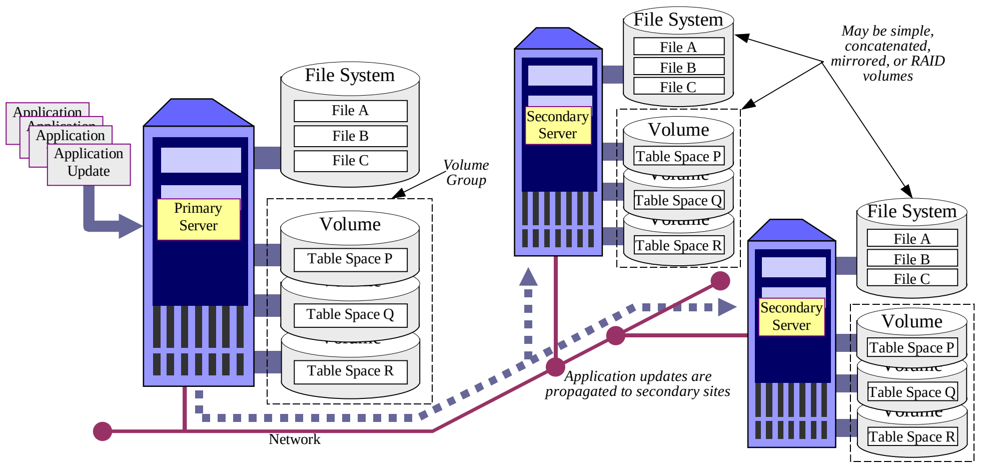


Figure 10: File and Volume Replication

In Figure 10, applications run on the primary server and update both files in a file system and database tables stored in a volume group (with or without a file system). Replicas of both the file system and the volume group are maintained on two secondary servers. Figure 10 illustrates three key points about replication:

- Either file systems or groups of volumes (containing one or more file systems) can be replicated. File system replication is useful for the many administrative and miscellaneous files that are an inevitable part of complex e-business applications. Volume group replication is usually more appropriate for databases stored in a small number of large files, possibly distributed across multiple file systems on multiple volumes.
- Replication is a *one to many* operation. There is a single source (the primary site) and one or more targets for each replicated file system or volume group. The primary site provides application read/write access to data. File system replicas can be mounted for application use. Practically speaking, volume group replicas cannot be used by applications while replication is active.

- Replication uses conventional hardware components and network connections. No special hardware is required to replicate data, nor must communication links be dedicated to replication (although it is recommended to have dedicated links to maximize throughput and application responsiveness.) Primary and secondary data replicas need not be stored on identical disks, although replicated volumes should have identical capacity.

Replication is a layer in the data access and management stack (Figure 1, page 10). Since replicated objects are file systems or volumes, all foundation features including availability enhancements like mirroring and performance enhancements like Quick I/O can be used in conjunction with data replication.

Replication is policy-based. System administrators can set policies for replication to meet business requirements. Replication policies include:

- which file systems or volume groups at the primary site are to be replicated,
- which file systems or volumes at secondary sites are replication targets,
- what degree of synchronization with the primary site must be maintained, and,
- how temporary failures such as network outages should be handled.

Once policies have been set, replication is automatic, requiring no system administrator intervention until an exceptional event such as a site disaster occurs.

While both file system and volume group replication accomplish the same basic goal—maintaining identical data on multiple computers—there are differences between the two that stem primarily from the nature of the replicated objects. These differences influence the ideal applications for each type of replication.

File System Replication

File system replication is *synchronous*. This means that application writes at the primary site are not complete until data has been written at all secondary sites. The mechanics of synchronous replication are discussed in the following section, however the net effect is that application writes to replicated file systems can take significantly longer to execute than writes to local file systems. File system replication is therefore generally unsuitable for applications with high update rates.

Since file system replication keeps data at secondary sites completely up-to-date, primary and secondary copies of data are never different, so there is no need to log updates to recover from communication or secondary site failures. While this minimizes overhead I/O for replication, there is a disadvantage—a loss of communication between primary and secondary sites requires a lengthy resynchronization process.

After a network outage, primary and secondary file replication sites must determine whether their respective copies of each replicated file are identical. They do this by independently computing a checksum on each file. If two sites' checksums for a file differ, the secondary site's version is replaced with a copy from the primary site.

Compared to log playback, this recovery technique is time-consuming. With file system replication, recovery time has been sacrificed in favor of minimal overhead during normal operation. File system replication is therefore best suited for more reliable networks in which outages that would require re-synchronization are very infrequent.

File system replication is typically very suited for:

➨ ***content distribution***: Some organizations maintain data at a central (primary) location and publish it for use at multiple secondary locations. Web pages, price lists, product specifications, and other documents used at multiple e-business locations are prime examples of this kind of application.

File system replication has another advantage—since quite a lot is known about the replicated data objects (e.g., whether a file is in use), file system replication can be bi-directional. Applications at a secondary site can access files in a replicated file system. Under some circumstances, files modified by applications at the secondary site can be replicated back to the primary site. This is not the case with volume group replication, which is strictly unidirectional.

Volume Group Replication

With volume group replication, blocks written to primary site volumes are replicated on volumes at one or more secondary sites. Since there is minimal context associated with volume replication, *asynchronous* replication is possible. The mechanics of asynchronous replication are discussed in the next section, however for applications, the key advantages are improved application performance and fast recovery from network outages.

During a network outage, the primary volume replication site logs all updates to replicated volume groups. After network recovery, the block updates logged at the primary site during the outage are sent to secondary sites. Volume group replication is therefore the more suitable option when the network between the primary and secondary replication sites is less reliable.

The nature of volume group replication makes it essentially impossible for applications at the secondary site to use a replica while replication is active. This makes volume group replication most suitable for:

➨ ***disaster recovery***: A *disaster recovery site* is a secondary data center located far enough from the primary data center that it can continue operating if some range of potential disasters occurs. (For example, a disaster recovery site might need to be further from a primary site in an earthquake prone area than would be the case in a geologically stable area). Volume groups at the primary site can be replicated to a disaster recovery site. If a primary site disaster occurs, applications can be quickly restarted at the disaster recovery site, and processing can resume using the (up to date) data replica.

With volume replication, the replication manager has no information about the nature of updated blocks—whether they hold file data, file system metadata, database pages, or other objects. Without such information, the replication manager cannot synchronize primary and secondary site updates. Volume group replication is therefore strictly unidirectional—blocks are copied from the primary site to one or more secondary sites, but not the reverse.

Synchronous Replication

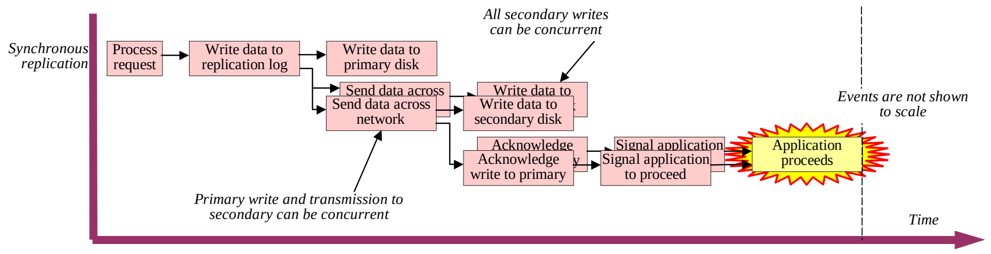
A secondary replica is said to be *up-to-date* if volume group contents at the secondary site are identical to their counterparts at the primary site. For a secondary site to be up-to-date at every instant, all updates would have to be replicated *synchronously* across all sites. That is, each application update would have to be written both to primary volumes and to the corresponding volumes at all secondary sites before the application could be allowed to proceed.

The number of sequential operations required to synchronously replicate data could result in unacceptably long application response times. VERITAS volume replication uses an optimization to improve application response time without sacrificing the goals of synchronous data replication. An application that writes to a synchronously replicated volume is allowed to proceed as soon as its write has been:

- logged at the primary site, and,
- transmitted to all secondary sites.

Figure 11 is a time line for this replication architecture, showing actions that can occur in parallel. Compared to a write to a local non-replicated volume, a write to a synchronously replicated volume takes longer by

- the local logging time (a disk I/O time), and,
- a round trip message time to the most distant (in time) secondary site.



Synchronous replication
Process request
Write data to replication log
Write data to primary disk
All secondary writes can be concurrent
Send data across network
Write data to secondary disk
Acknowledge write to primary
Signal application to proceed
Application proceeds
Events are not shown to scale
Primary write and transmission to secondary can be concurrent
Time

Figure 11: Synchronous Volume Group Replication Time Line

Using the replication algorithm illustrated in Figure 11, data is secure against:

- primary site disaster, because a copy exists at each secondary site, and,
- secondary site or communications link failure, because all updates are logged at the primary site.

Even with this optimization, peaks in application update rates, momentary network overloads, or simply a large number of secondary sites can make synchronous replication impractical from an application performance standpoint. VERITAS volume replication supports an *asynchronous* replication mode for these circumstances.

Asynchronous Replication

With asynchronous replication, applications are allowed to continue execution as soon as their write requests have been logged at the primary site. Transmission and writing to volumes at secondary sites occurs asynchronously, usually after the application has been notified that its write is complete. Figure 12 illustrates the difference in application timing between synchronous and asynchronous replication.

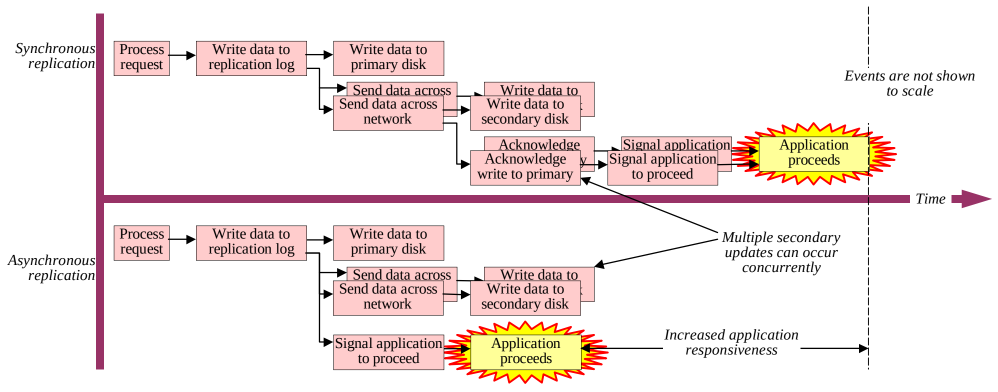

Synchronous replication
Process request
Write data to replication log
Write data to primary disk
Send data across network
Write data to secondary disk
Acknowledge write to primary
Signal application to proceed
Application proceeds
Events are not shown to scale
Time
Asynchronous replication
Process request
Write data to replication log
Write data to primary disk
Send data across network
Write data to secondary disk
Multiple secondary updates can occur concurrently
Signal application to proceed
Application proceeds
Increased application responsiveness

Figure 12: Performance of Synchronous and Asynchronous Replication

As Figure 12 illustrates, asynchronous replication reduces *latency*, or end-to-end application write request execution time. The more important consequence of asynchronous replication, however, is that momentary network overloads do not stall applications or cause their write requests to fail. Applications that write to asynchronously replicated volumes do not experience slow down or stall due to network overload, because their execution proceeds independently of communication link and secondary site activity.

The replication manager sends writes requests from the primary site log to secondary sites as quickly as network and secondary site loading permit. If a network overload is transient, it eventually clears, and secondary sites become up-to-date. If the network overload is chronic, the primary log of not-yet-replicated writes will grow, and applications will eventually stall. Asynchronous replication can buffer short-term network overload, but it is not a substitute for inadequate steady state network bandwidth.

The advantages of asynchronous replication are generally better application response (compared to synchronous replication), tolerance for momentary network overloads, and fast recovery after secondary site crashes or network outages. The disadvantage is that there can be brief periods of time during which secondary volumes are not completely up-to-date. If a secondary computer crashes or a communication link fails when in this state, data from the primary log is transmitted and written after recovery. If, however, there is an unrecoverable disaster at the primary site, and primary log contents cannot be retrieved, secondary site recovery must proceed with slightly out-of-date data.

To limit this exposure, a system administrator can designate a maximum number of writes by which secondary sites are allowed to be out-of-date. When this limit is exceeded, application writes at the primary site stall (no completion signal is given) until the amount of unsent data has fallen below the permissible threshold. This threshold bounds the amount by which replicated data can be out-of-date.

Even though it does not guarantee perfect consistency, asynchronous replication is often an operational necessity for performance reasons. Either momentary update overload or heavy network loading from other sources can increase application response times unacceptably. Asynchronous replication eliminates most of the actions required to replicate data remotely from application response time, and can be the difference between replication being operationally practical or not.

III. Clustering

Keeping E-Business Applications Available

The *Processing* in E-Business Data Processing

Thus far, this paper has discussed storage and data management for e-business—the presentation of highly available data to applications at adequate levels of performance to meet business needs. To meet e-business availability and scalable growth requirements, however, one must also provide highly available data *processing*. An e-business must deal with questions like:

- How can continuous application service be provided, even if systems fail or data centers become incapacitated?
- How can application performance scale as an e-business grows, and the load on its computing resources increases?
- How can multiple related systems be centrally managed at reasonable cost?

Clustering coordinates the operations of applications on several servers to enhance application availability and scalability.

Clusters Defined

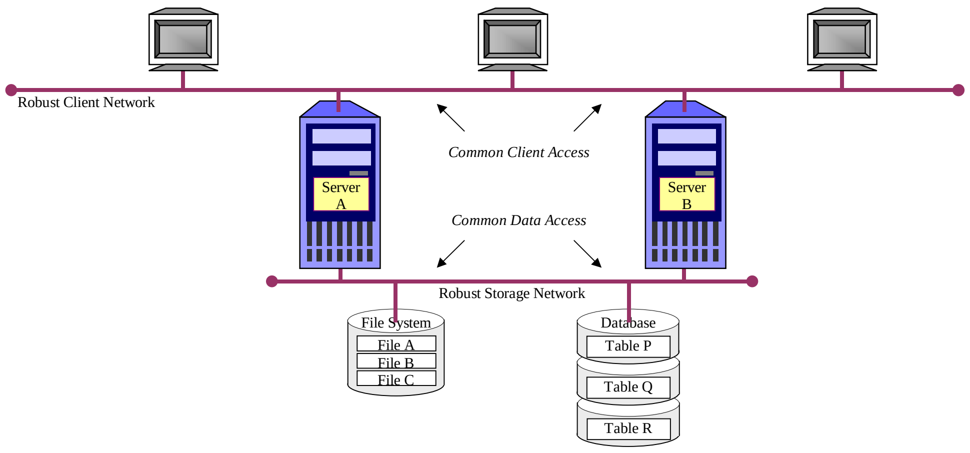


Figure 13: Basic Cluster Model

A cluster is a set of interconnected computers and storage whose operation is coordinated to achieve some beneficial effect. Figure 13 illustrates the basic features of a cluster. Clustering is most often used to enhance application availability and scalability. Some clustering technologies add the concept of a *single system image*, allowing multiple computers to be managed from a single point as if they were a single system.

Clusters have been conceptually attractive to computer users for some time because they can potentially solve some of the classic problems in computing:

- If a server fails or an application crashes, another server in the cluster can take over its workload, since all servers are connected to the same data storage devices and the same clients.
- If the demands of an application become too great for the existing servers, additional servers can be added, and the workload divided among the new, larger complement of servers.
- If network links fail, clients can use alternate paths to access data and continue to operate.
- If an entire data center becomes incapacitated, remote computers belonging to the cluster can take over its workload and continue processing using a replica of online data.

All of these benefits can be realized to one degree or another with clustering.

Applications and Clusters

Since clustering essentially enhances the behavior of applications, a reasonable starting point is to examine an application as the cluster server sees it—as a *service* provided by a group of related system *resources*. For example, a web server application might consist of:

- disks on which the web pages to be served are stored,
- a volume built upon the disks,
- a file system using the volume,
- a database whose tablespaces are files and whose rows contain page pointers,
- the network interface card (NIC) or cards used to export the web service,
- one or more IP addresses associated with the network card(s), and,
- the application program and associated code libraries.

From a cluster standpoint, there are two significant aspects to this view of an application service as a collection of resources:

- If a service is to run on a particular server, all of the resources it requires must be available to the server.
- The resources comprising a service have interdependencies; that is, some resources (e.g., volumes) must be operational before other resources (e.g., the file system) can be made operational.

Figure 14 illustrates a resource dependency graph for a cluster service.

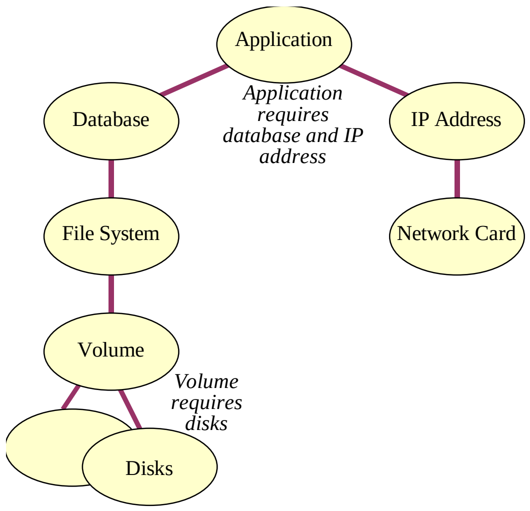


Figure 14: Cluster Service Resource Dependency Graph

In Figure 14, the lower (*child*) nodes represent resources required by the upper (*parent*) nodes to function. Thus, the volume requires that the disks be online, the file system requires that the volume be active, and so forth. The application program itself requires two independent resource subtrees to function—a database and an IP address for client communications.

The resource groups discussed thus far are known as *failover groups*. The resources in a failover group can only be online on one of a cluster's computers at any instant. This prevents two computers from attempting to run the same application against the same data. The cluster engine also recognizes other types of resource dependencies. For example, it may be necessary to prevent two services from running at the same time, as with test and production versions of the same application. These relationships can be expressed in *resource group* graphs that specify dependencies among entire services. The cluster engine manages these dependencies. For example given, the cluster engine would prevent both test and production applications from running at the same time.

For application scaling, the cluster engine also recognizes *parallel* resource groups. The resources in a parallel group can be online on more than one server at a time. This allows applications that require more than the capacity of a single server to run as multiple instances, each on a separate server. In a parallel resource group, it is the application's responsibility to coordinate concurrent accesses to shared data from multiple application instances on multiple servers in a cluster.

Accommodating Different Types of Resources

The actions required to bring a resource online or take it offline differ significantly for different types of resources. Bringing a disk online, for example, might require a SCSI command to spin it up, whereas bringing an Oracle database online would require starting the database manager process and issuing the appropriate Oracle `startup` command(s) to it. From the cluster engine's point of view the same result is achieved—making the resource available.

Each type of resource supported in a cluster is associated with an *agent*. An agent is an installed program registered with the cluster engine. An agent has three *methods*, or entry points at which it can be called:

- ***online***, which is invoked by the cluster engine when the resource is to be brought online,
- ***offline***, which is invoked by the cluster engine when the resource is to be deactivated, and,
- ***monitor***, which is invoked to test the operational state of the resource.

The cluster engine invokes an agent's methods at appropriate times to start and stop application services, and to perform *failover*. Each agent method has a list of parameters, called *attributes*, which are supplied when it is invoked by the cluster engine.

Since the structure of cluster resource agents is straightforward, it is relatively easy to develop agents as additional cluster resource types are identified. For example:

- an ***Oracle process agent*** that issues **`startup`** and **`shutdown`** commands, and,
- a ***SqlNet Listener agent*** that starts and stops the Oracle client listener.

Both of these agents' monitor entry points run Oracle **`Monscript`** to monitor database state if their respective resources are running.

Using Clusters

Failover

Perhaps the main use of clusters in e-business is to enhance application service availability. Application services can fail because a critical resource (e.g., the application program or the host bus adapter connecting the server to application data) fails, or because the server on which the application is running fails. Whatever the reason, when the cluster engine detects the failure (through the absence of the service resource group's monitored heartbeat signal), it initiates a failover operation to restart the application on another computer. When configuring a cluster, the administrator provides an ordered list of computers eligible to run each application service. Specifying the affinity between failover service resource groups and computers enables *cascading failover*. Referring to Figure 15, if Cluster Server A is designated as the preferred service resource group for an application, then that application will run on Server A when initially started. If Server A were to fail, the service group would restart on whichever of the servers was designated as the preferred failover server. If the preferred failover server were to fail, a secondary failover server would take over, and so forth. If the physical configuration permits, every server in a cluster can be designated as a potential failover server for a given service group.

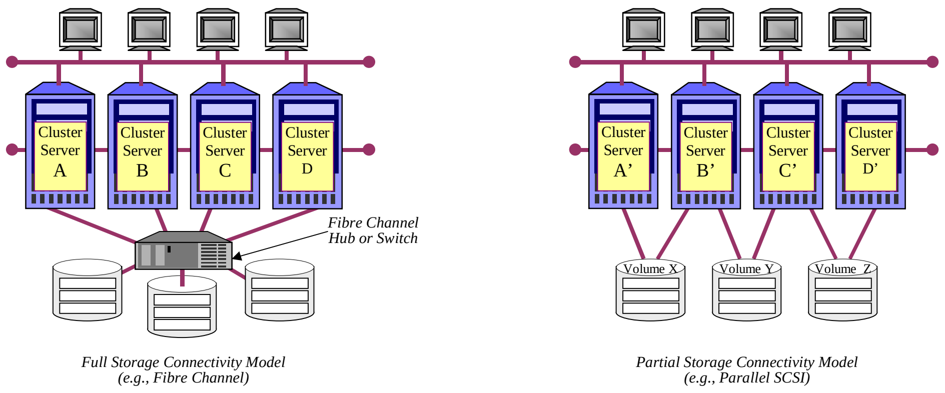


Figure 15: Full and Partial Storage Connectivity in a Cluster

In order for a server to run an application service, it must have access to all of the service's resources. Two types of resources, storage objects and process objects (whose images reside on storage objects) may limit the computers on which a given service can run, because they may not be physically connected to all of the computers in the cluster. Figure 15 illustrates two cluster connectivity models:

- ***full storage connectivity***, in which every computer in the cluster has a direct physical connection to all storage devices in the cluster.
- ***partial storage connectivity***, in which not all computers are directly connected to all storage devices.

In a hardware configuration with full storage connectivity, any computer in the cluster can be designated as the failover target for an application service group. Thus, in the cluster represented on the left of Figure 15, an application service that normally runs on Server A could fail over to Server B, Server C, and Server D, in that order (or any other order). In the cluster represented on the right of Figure 15, however, an application service that normally runs on Server A' could fail over to Server B', but not to the other servers, because they have no access to its data or application program images.

This example illustrates the care that a system administrator must exercise in configuring clusters when the hardware complement provides only for partial storage connectivity. A hardware solution that offers full storage connectivity, such as a Fibre Channel SAN is preferable from a flexibility and growth standpoint.

Application Scaling

Figure 16 illustrates a cluster configured to run a parallel application on more than one server concurrently.

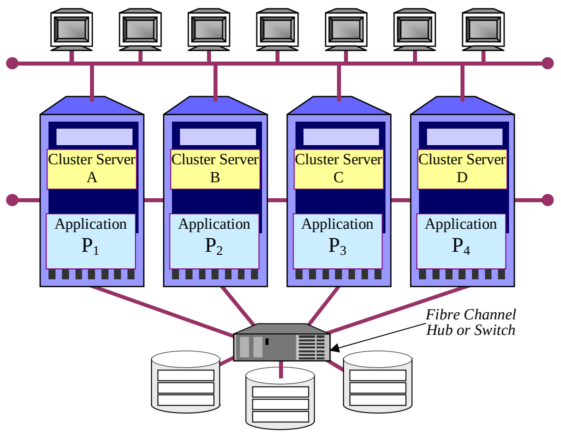


Figure 16: A Parallel Service Resource Group

Figure 16 represents a cluster running an instance of application service P on each of its four servers. The cluster engine recognizes parallel service resource groups and automatically starts the four application instances when the cluster initializes. If, however, multiple instances of an application are to share data, as Figure 16 implies, then the application or database management system must incorporate some mechanism to prevent multiple instances of itself from interfering with each other.

Uncoordinated simultaneous access to data from multiple applications can result in data corruption in a variety of ways. Suppose, for example that an online sales application is updating a running total of daily business volume. Each time a sale was made, the daily total would be read, the amount of the sale added to it, and the result written back to storage. Figure 17 illustrates one of several ways in which two or more instances of this application running on separate computers in a cluster might lead to a corrupted daily sales total.

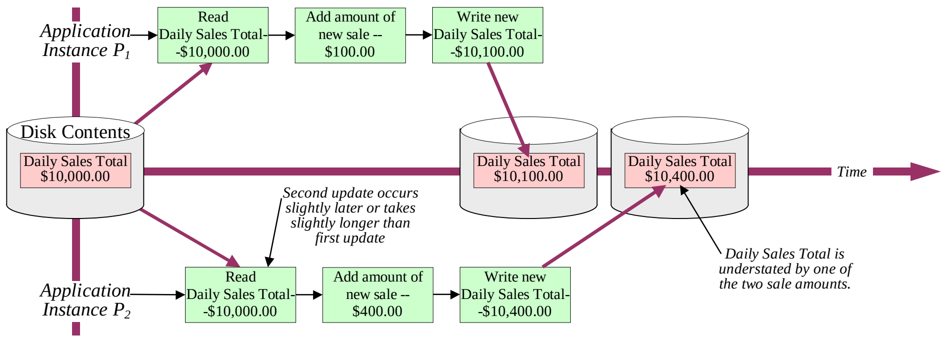

Application Instance P_1
Read Daily Sales Total--$10,000.00
Add amount of new sale -- $100.00
Write new Daily Sales Total--$10,100.00
Disk Contents
Daily Sales Total $10,000.00
Daily Sales Total $10,100.00
Daily Sales Total $10,400.00
Time
Second update occurs slightly later or takes slightly longer than first update
Daily Sales Total is understated by one of the two sale amounts.
Application Instance P_2
Read Daily Sales Total--$10,000.00
Add amount of new sale -- $400.00
Write new Daily Sales Total--$10,400.00

Figure 17: Data Corruption Scenarios with Simultaneous Application Access

The data corruption problem illustrated in Figure 17 occurs because the update processes are unaware of each other. For correct operation, both of the read-modify-write update sequences must be *atomic*; that is, each sequence must be the only thing happening to the Daily Sales Total record for the duration of the sequence. If the application instances making these updates were running on the same computer, a file system that allowed *multiple simultaneous writers* would be required for correct operation. Such file systems include a *local lock manager* that reserves access to ranges of bytes in a file to prevent simultaneous updating by multiple processes.

If the processes represented in Figure 17 are running in different computers, then a *distributed lock manager* is required. While a local lock manager maintains state in memory data structures, a distributed lock manager exchanges messages to indicate which data objects are locked or released. Multiple instances of a file system running in separate computers and using a distributed lock manager to coordinate accesses to data are collectively called a *distributed* or *cluster file system*. Database management systems, such as Oracle Parallel Server, support similar functionality with embedded distributed lock managers for locking database objects stored in raw devices today and in clustered file systems in the future.

Clusters and Disaster Recovery

Clustering technology can be combined with storage replication to further automate recovery from site disasters. Figure 18 illustrates one possible disaster recovery scenario using a combination of replication and clustering technologies.

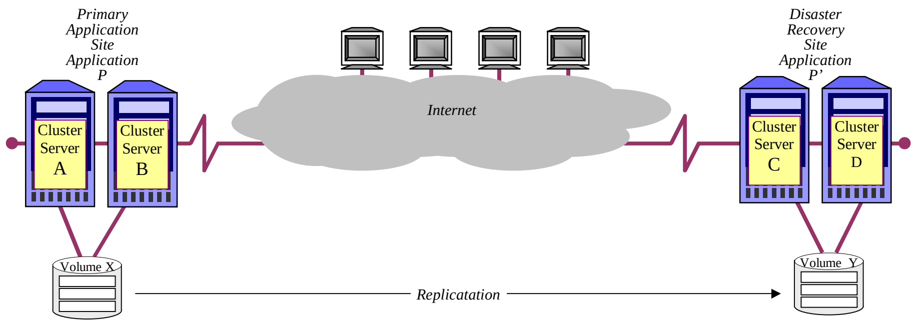


Figure 18: Combining Cluster and Replication Technologies

In Figure 18, Servers A, B, C, and D comprise a cluster. In addition, the data from Volume X is replicated to Volume Y across a WAN. Application service P can run on Server A, and fail over to Server B. Similarly, Application P' can run on Server C, and fail over to Server D. Application P' would consist of:

- a script to disengage Volume Y as a replication secondary and remount it as a read/write volume,
- a script or program to perform any necessary application dependent data checking required before application P can be restarted, and,
- the same program image(s) used by application service P.

In cluster server terms, application service resource groups P and P' would be configured to have an offline dependency as described earlier. The cluster engine would only bring service group P' online when service group P went offline. If Server A or B failed, the other server at that site would fail over application service P, and service group P' would not be activated. If, however, a disaster incapacitated the entire site, then heartbeats from both Server A and Server B would be missed. Cluster monitoring mechanisms at Servers C and D would declare service group P to be offline, which in turn could activate application service P'. After the restart housekeeping, the alternate instance of the same application image from service group P would resume services to clients from the alternate site.

Clustering in an E-Business Context

In recent years, the data processing requirements of e-business have led to a bewildering variety of large system topologies. Most of these include clustering technology in one or more forms. Figure 19 illustrates a large system with clustering at three (arguably, four) different levels.

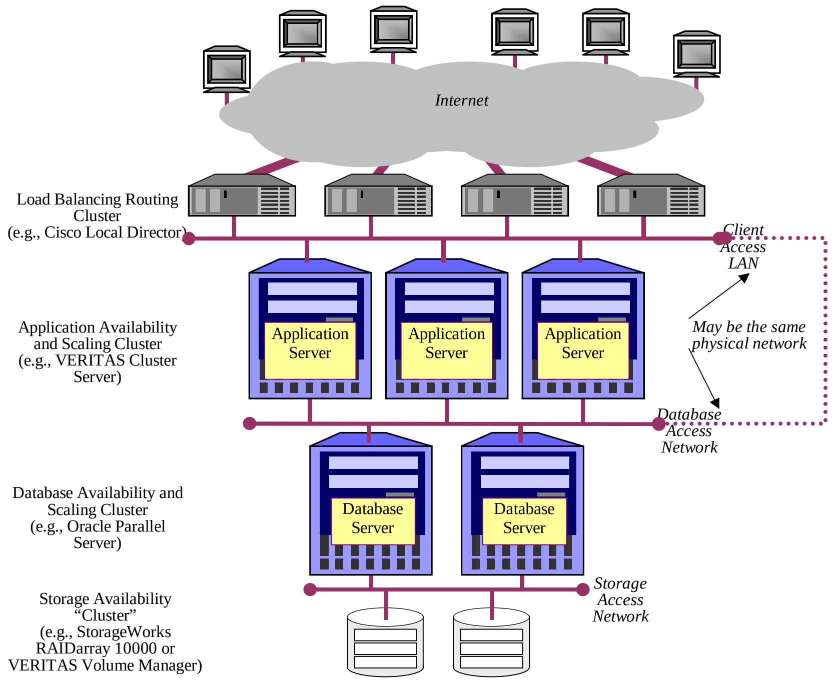

Internet
Load Balancing Routing Cluster (e.g., Cisco Local Director)
Client Access LAN
Application Availability and Scaling Cluster (e.g., VERITAS Cluster Server)
Application Server
Application Server
Application Server
May be the same physical network
Database Access Network
Database Availability and Scaling Cluster (e.g., Oracle Parallel Server)
Database Server
Database Server
Storage Access Network
Storage Availability "Cluster" (e.g., StorageWorks RAIDarray 10000 or VERITAS Volume Manager)

Figure 19: Large-Scale E-Business Data Center Model

As client requests enter the system represented in Figure 19, they are handled by one of a set of cooperating network routers, using a technology such as Cisco Systems' *Local Director*. Local Director and systems like it balance incoming network load across a number of application servers. Most cooperating routers provide failover in the event of a single router failure.

At the application level as illustrated in Figure 19, cluster server technology can provide application scaling to multiple servers, with the surviving servers picking up the workload in the event of application server failure. The coordination of shared access to data is managed through the use of a third tier in the computing hierarchy—a database server tier, running Oracle Parallel Server, for example. Oracle Parallel Server coordinates access to data from multiple sources (the cluster of application servers) across multiple servers (the Oracle Parallel Server database servers), assuring database consistency.

Finally, in systems of this size and complexity, failure tolerant scalable storage subsystems at the physical storage level are generally regarded as a requirement. With the I/O coordination, mirrored cache, and controller failover capabilities typically offered by these subsystems, it is almost appropriate to regard them as storage clusters in their own right. When combined with host based volume management and data replication technology, failure tolerant storage subsystems make up a truly robust data storage environment.

While systems of this sort appear complex, they provide the kind of completely scalable, highly available client-server computing that is vital to e-businesses as they emerge from the startup phase into the enterprise arena.

IV. Data Management

Non-Disruptive, Efficient Protection of Information Assets

What Is Data Management Anyway?

As a primary e-business asset, online data must be managed. While the term *data management* has many meanings, VERITAS uses the term to mean managing data so that:

- ➨ e-business operations can resume as quickly as possible after a computer, storage, software, or site failure,
- ➨ data gets to where it’s needed, when it’s needed by the business,
- ➨ regulatory and business policy record retention requirements are met.

As has already been emphasized, the primary goal of e-business information services departments is keeping online data available for partner access. Data management in an e-business context therefore means meeting these goals in an environment of 24x7 online databases.

Behind the seemingly simple action of copying and moving data objects there are significant technical challenges:

- ➨ designing and implementing a policy that gets data to where it is needed when it is needed, even if errors occur,
- ➨ keeping track of which data is at which location(s), for example, which backups are on which tapes and where those tapes are located,
- ➨ guaranteeing the self-consistency of collections of data objects as they are moved or copied,
- ➨ minimizing the *service outage time* during which data objects are unavailable to applications because they are being moved or copied, and,
- ➨ determining when changes in management policy would be beneficial, for example, when backups should be more frequent, or when copies of product data or price lists should be replicated at regional offices to reduce network traffic.

Backup: The Core of Data Management

Backup is central to any data management architecture. A backup is a copy of a defined set of data, ideally as it exists at a specific point-in-time.[4] Backup copies are stored separately from operational data, usually on tape or other removable media, and can be stored outside the data center so that they are likely to survive events that destroy or corrupt operational databases. Backup copies can be:

- kept at the data center, so that if a storage device, system, application, or human failure destroys vital online data, the e-business can restore its state as of a recent point in time. From that point, database logs (presumably stored separately) can restore a nearly up-to-date business state.
- moved to one or more other sites, to provide similar protection against environmental events, such as fire or flood, that destroy entire data centers. With a backup copy of its operational databases available, an e-business can resume operations as soon as alternative computing facilities are available.
- made unalterable (for example, burned onto CDROM or other write-once storage technology) to provide the durable business records required for regulatory and business policy purposes when the data is no longer required to be online.

Backup Policies for E-Business

All data required to operate an e-business must be backed up. On the other hand, backup is a very resource intensive operation, so there is a natural desire to minimize the impact of backup on operations. System administrators express the trade-off between these two conflicting objectives in a *backup policy*. A backup policy is a set of rules that specifies:

- ***what*** data objects are to be backed up,
- ***when*** the data is to be backed up,
- ***where*** the data is to be backed up.

The following paragraphs describe how backup management components automatically execute backup policies.

What Data Objects to Back Up

Deciding ***what*** data objects to back up requires knowledge of both business policy and system operations. Effective backup policies distinguish seldom-changing data from rapidly changing data, and back up the former less frequently than the latter.

4 There are "fuzzy" file and database backup techniques that create copies of changing data with limited currency and consistency guarantees. These can be used to restore databases after a failure, however they have limited use as business records.

Data to be backed up can be specified as a list of files. In large or active environments, it is usually more appropriate to specify that the entire contents of one or more directory trees be backed up. This makes it unnecessary to track file additions and deletions from the backup policy specification.

When to Back Up

Deciding ***when*** to back up also requires both business and operations knowledge. System administrators must balance between acceptable backup age (how many hours of business records must be recreated from means other than backups) and the impact of backup resource consumption on operations. If resources were not a consideration, the obvious backup policy would be to back up all online data constantly—to copy each object in its entirety each time it changed.

Resources *are* a consideration, however. Constant backup would consume significant processing, I/O, and network capacity, as well as large amounts of storage and catalog space, adversely affecting cost and online application performance. Backup schedules are therefore usually designed to minimize impact on online applications. For businesses that have predictable busy and idle periods, backups are typically scheduled during idle time. As operations shift toward e-business, however, organizations cannot rely on predictable idle periods. Techniques must be found to minimize the resources consumed by backup so it can coexist with online applications.

Where to Back Up Data

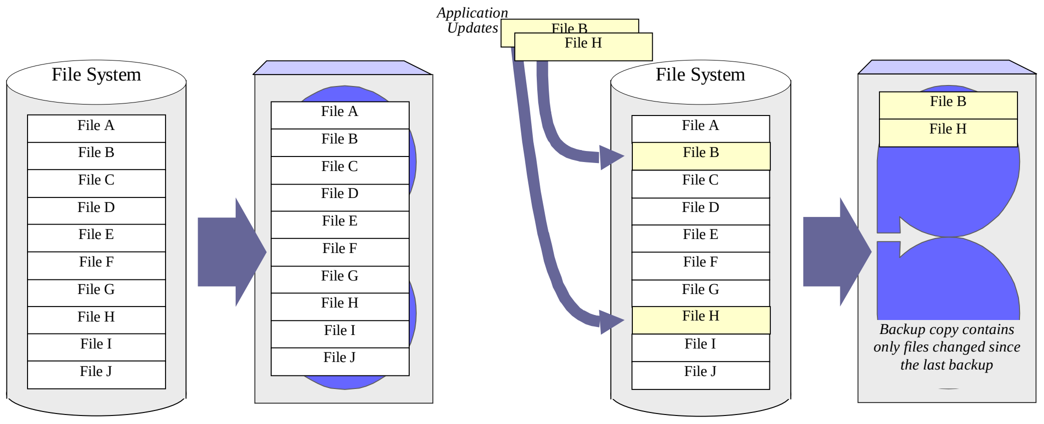
On the surface, ***where*** to back up data appears to be a simple question. The backup client is the data source. The destination is one of (possibly several) media servers. The choice of media server might differ depending on business cycle, equipment availability, or other considerations. A master backup server, like the one from VERITAS, keeps track of executing backup jobs on each client, and chooses a media server to receive backup data based on relative loading or backup device availability.

A media server generally chooses the specific backup device (e.g., tape drive) according to policy guidelines set by the system administrator. A system administrator can organize backup devices into groups, and associate each scheduled backup job with one device group. The media server may choose any available device within a group for a particular backup job.

Media (tapes or optical disks) are managed similarly. Available media are organized as *pools*, and each scheduled backup job is associated with a media pool. A *backup manager* chooses available media from a pool using an algorithm designed to equalize media usage (and therefore wear). A media manager can also maintain media cleaning and retirement schedules and keep track of media location.

Full and Incremental Backup

In most e-business environments, only a small fraction of online data changes between successive backups. In file oriented systems, only a small percentage of the files change. *Incremental backup* technology uses this fact to minimize backup resource requirements. An incremental backup is a copy of only the data or files changed since the last backup. A backup agent determines what objects have changed by inspecting file system metadata, and copies only those objects. Figure 20 illustrates the difference between a full and an incremental backup.



Full Backup (baseline) *Incremental Backup (changes to baseline)*

Figure 20: Full and Incremental Backups

Incremental backup does not replace full backup; it augments it. An incremental backup contains data objects that have changed since some point-in-time for which a full backup is available. To restore a complete file system from incremental backups, a full backup must first be restored as a baseline. The incremental backups are then restored in order, overwriting changed files in the baseline. Incremental backup reduces the frequency of performing time-consuming full backups.

If only a small percentage of the files in a large file system have changed since the last backup, only a small percentage of the data must be backed up. Incremental backups typically complete much faster, and consequently have less impact on online operations, than full backups.

When an incremental backup policy is in use, the master backup server keeps track of the sequence of full and incremental backups. For restoring individual files, the master backup server identifies the latest backup copies. For restoring complete file systems, the master backup server guides the system administrator through the right sequence of tape mounts for the required full and incremental restores.

Different Types of Incremental Backup

There are two generally recognized forms of incremental backup. With a *differential backup*, all files modified since the last backup of any kind are copied. Thus, with a policy of weekly full backups and daily differential backups, the most up-to-date file system restore is accomplished by restoring the newest full backup, and then each of the newer differential backups in order by age (oldest first). The later in the week, the longer a restore operation will be.

A *cumulative backup* copies all files modified since the last *full* backup. To restore a file system from cumulative backups, only the newest full backup and the latest cumulative incremental backup are required. Restoring file systems is simpler and faster, but at the cost of lengthening backup times as the time since the last full backup increases.

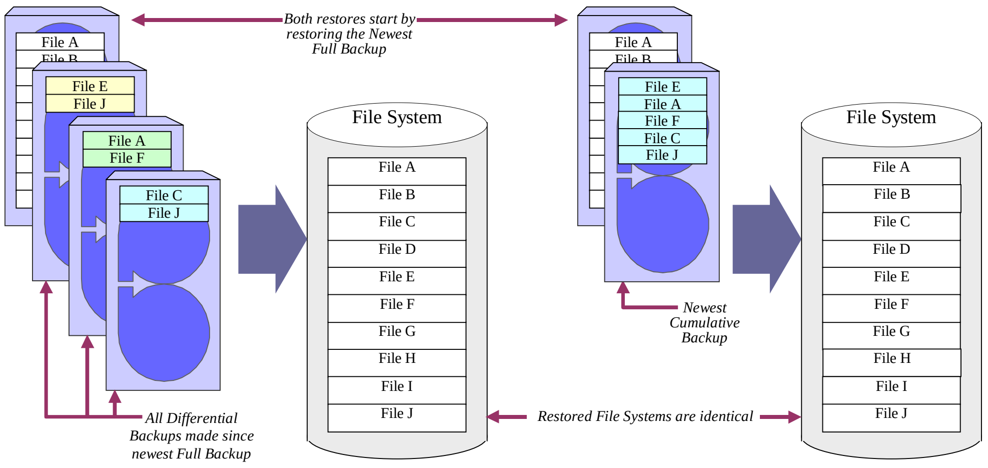


Figure 21: Restoring a File System from Differential and Cumulative Backups

Full, cumulative, and differential backups can be combined to balance the impact of backup on operations against the time required to restore a full file system or database. Table 3 illustrates a backup scheduling strategy in which full, differential, and cumulative backups combine to balance backup time and restore complexity.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Type of Backup	Full backup	Differential Incremental	Differential Incremental	Cumulative Incremental	Differential Incremental	Differential Incremental	Differential Incremental
Data in backup copy	Full database as it stood on Sunday	Files changed since Sunday's full backup	Files changed since Monday's backup	Files changed since Sunday's full backup	Files changed since Wednesday's backup	Files changed since Thursday's backup	Files changed since Friday's backup
Full Database Restore Procedure	Restore Sunday's backup	Restore Sunday's backup and Monday's differential	Restore Sunday's backup, Monday's and Tuesday's differentials	Restore Sunday's backup and Wednesday's cumulative	Restore Sunday's backup, Wednesday's cumulative, and Thursday's backup	Restore Sunday's backup, Wednesday's cumulative, Thursday's and Friday's differentials	Restore Sunday's backup, Wednesday's cumulative, Thursday's, Friday's, and Saturday's differentials

Table 3: A Sample Weekly Backup Strategy

System administrators can define automatic backup schedules like that illustrated in Table 3 to meet business needs. With robotic tape libraries, scheduled backup can be completely automated. No system administrator or computer operator action is required once a policy has been set.

Backup and Databases

Conventionally, database management system vendors have integrated point-in-time backup facilities with their products. While different vendors' mechanisms differ in detail, the effect is similar to the file system snapshot facility described on page 21. Database activity must cease momentarily so that a backup can be initiated. After a backup is initiated, each modification to a database object causes a copy of the old contents of the object to be saved. When the backup program reads data, these before images are returned. (When any other program reads data, the current object contents are returned). A backup made in this way represents the contents of the database at the point-in-time at which the backup was initiated. This technique, often called *hot database backup*, is well accepted and widely used.

A consistent point-in-time backup of an online database can be created with minimal overhead using a VERITAS file system facility called S*torage Checkpoints*. Storage Checkpoints are functionally similar to the file system snapshots described on page 21 *ff.*, but are only accessible to VERITAS backup software. Each Storage Checkpoint is a point-in-time image of one or more file systems containing a database. Storage Checkpoints use the same *copy-on-write* technique as snapshots (described on page 22) to minimize storage space requirements. Storage Checkpoints differ from snapshots in that:

- They are persistent (i.e., they still exist after a reboot).
- They use the file system free space pool.
- They are aware of each other to backup at the block instead of file level.

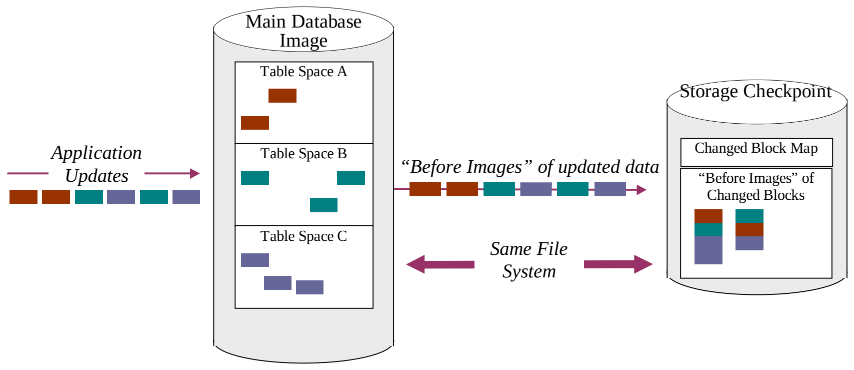


Figure 22: Storage Checkpoints

Like snapshots, Storage Checkpoints should be initiated when a database is quiescent and its disk image is *consistent*. At such instants no transactions are in progress and all cached data is reflected on disks. Communication with Oracle insures Storage Checkpoint consistency. Storage Checkpoint creation begins with a request to Oracle for a momentary database shutdown. When Oracle reports that the database is quiescent, a Storage Checkpoint is initiated. After Storage Checkpoint initiation, which lasts a few seconds, Oracle is requested to restart the database for application use.

Storage Checkpoints initially occupy a small amount of storage space—enough for a *changed block map* of the file system containing the database. As applications write to the database, blocks are allocated to the Storage Checkpoint, before images of updated blocks are copied to them, and the changed block map is updated.

Storage Checkpoints can eliminate backup windows for databases. Both full and incremental database backups can be taken from a Storage Checkpoint. Moreover, since the backup software reads from different file system objects (the Storage Checkpoints) than database applications, one source of database overhead I/O is eliminated.

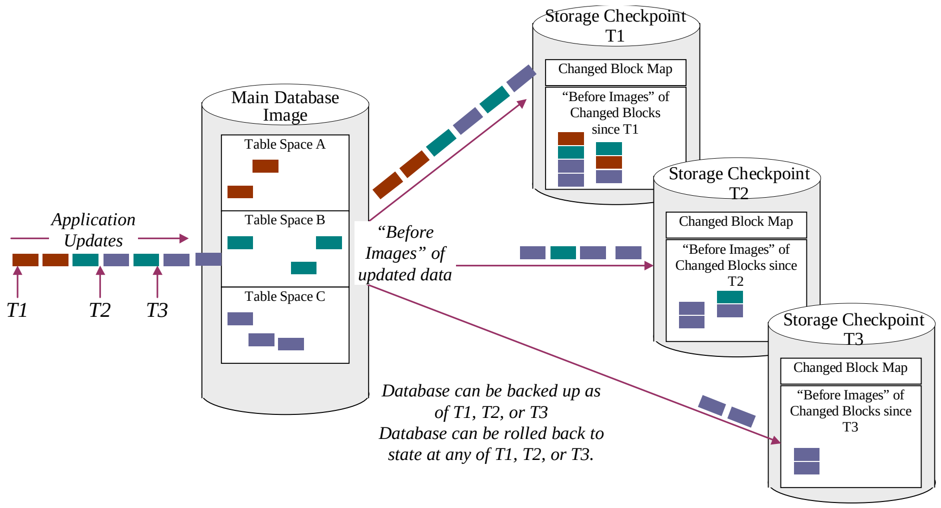

Storage Checkpoint T1
Changed Block Map
"Before Images" of Changed Blocks since T1
Main Database Image
Table Space A
Table Space B
Table Space C
Application Updates
T1
T2
T3
"Before Images" of updated data
Storage Checkpoint T2
Changed Block Map
"Before Images" of Changed Blocks since T2
Storage Checkpoint T3
Changed Block Map
"Before Images" of Changed Blocks since T3
Database can be backed up as of T1, T2, or T3
Database can be rolled back to state at any of T1, T2, or T3.

Figure 23: Multiple Checkpoints and Database Rollback

Multiple Storage Checkpoints can be maintained concurrently, as Figure 23 illustrates. While each Storage Checkpoint uses both storage capacity and I/O resources (when data is updated), this facility can give the administrator multiple choices. For instance, before images of changed blocks from a Storage Checkpoint can be written back to the main database. This facility can be applied to an entire database, to individual tablespaces, or to individual datafiles. Using this facility effectively "rolls back" the database or tablespace to the instant of Storage Checkpoint creation. It can be useful, for example, if an application error is discovered only after the application has been run against the database for a period of time.

Since Storage Checkpoints use file system free space, multiple Storage Checkpoints could cause unexpected space allocation failures. Rather than let this happen, the file system deletes Storage Checkpoints when space requirements cannot be met in any other way. Alternatively, the administrator can set a script to run when file system free space drops below a prescribed level. The script could take any number of corrective actions, such as extending the file system (and the underlying volume), or deleting Storage Checkpoints that are no longer needed.

Block-Level Incremental Backup

While they are extremely useful in file oriented systems, neither differential nor cumulative file-level incremental backup is particularly database-friendly. A typical database stores its tables in a small number of large files, most or all of which change frequently as the database is used. Thus, an incremental backup that copies changed files in their entirety is likely to include all of the database's tablespaces, even if only a miniscule fraction of the data in them has changed since the last backup.

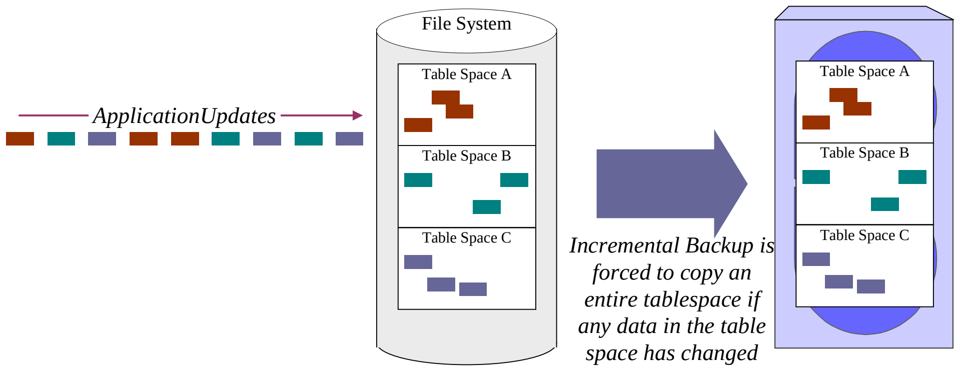
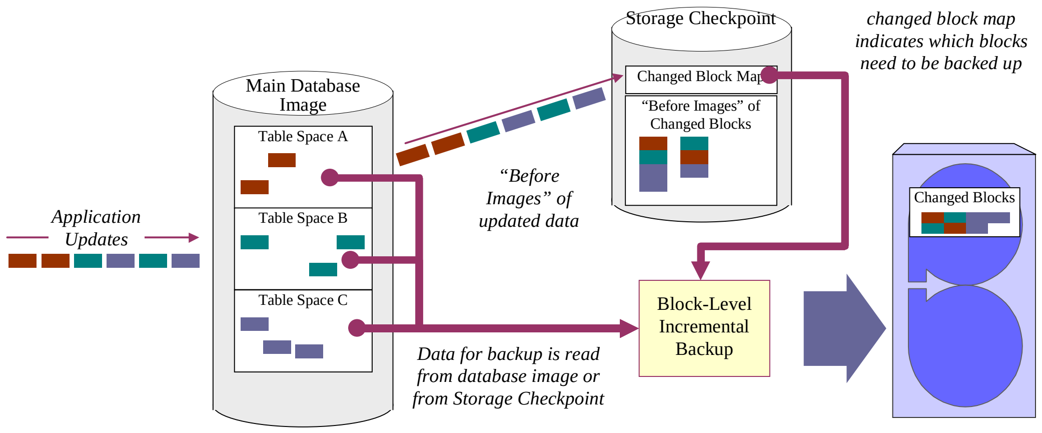


Figure 24: Limitation of Incremental Backups with Databases

The changed block map of a Storage Checkpoint, however, identifies the database blocks changed since the instant of checkpoint creation to create a *Block-Level Incremental (BLI) Backup*. Figure 25 illustrates BLI Backup.



Storage Checkpoint
changed block map indicates which blocks need to be backed up
Changed Block Map
"Before Images" of Changed Blocks
Main Database Image
Table Space A
Table Space B
Table Space C
"Before Images" of updated data
Application Updates
Changed Blocks
Block-Level Incremental Backup
Data for backup is read from database image or from Storage Checkpoint

Figure 25: Consistent Database Backup Using Storage Checkpoints

A BLI Backup contains the contents of only those database blocks modified since the instant of storage checkpoint creation. If only a small percentage of the database is updated in this interval, the BLI Backup image is correspondingly small. Compared to a full database backup, BLI Backup typically takes very little time and uses only small amounts of storage and I/O bandwidth.

Like file system incremental backups, BLI Backups must be relative to a full backup. To a restore a database from BLI Backups, a full backup made from the Storage Checkpoint on which the BLI Backups are based is required.

By greatly reducing the impact of an individual backup, BLI Backup encourages system administrators to schedule more frequent backups. Frequent BLI Backups reduce backup resource requirements (bandwidth and storage capacity) and enable databases to be restored to points-in-time closer to the instants of failure.

Backup Tactics

Multiplexed Backups

In a distributed e-business operation such as that illustrated in **Error! Reference source not found.**, several variables can affect the speed with which a backup job is accomplished:

- ***client load:*** An application server busy with other work may prevent backup client software from accessing data fast enough to keep the backup data path busy.
- ***network load:*** A network busy with application traffic may prevent a backup client from transferring data fast enough to keep a backup server or tape drive busy.
- ***backup server load:*** A backup server may be too overloaded with concurrent backup jobs (or other work, if it is doubling as an application server) to keep tape drives streaming.
- ***tape drive data transfer rate:*** Tape drive performance degrades significantly if data is not supplied fast enough to keep the drive *streaming* (i.e., with tape in motion and writing data). A small gap in the data stream supplied to a tape drive can result in a much larger interruption of data flow as the drive repositions itself.

Additionally, effective media utilization can be an important consideration. High capacity tape cartridges typically have two to four times as much capacity as a disk. A policy of frequent incremental backups adopted for business reasons may result in many small backup data sets. Each backup data set might only occupy a small fraction of a tape's capacity. Not only are underutilized media costly, but unnecessarily large libraries of media increase storage cost and the probability of handling errors.

The most visible method for minimizing the negative effects of performance differences along the backup data path and promoting is to *multiplex*, or interleave blocks of data from several backup jobs on a single tape. Such a policy can compensate for slow client data feeds, busy networks, and speed mismatches between network and tape drive. When multiple backup streams are interleaved, blocks from each stream are tagged with a job identifier, and written to tape in order of arrival at the backup server. With more data arriving at the backup server, tape streaming increases, improving backup throughput across the system. Since data from several jobs goes onto the same tape, media utilization increases. If a single file or file system restore from an interleaved backup job is required, the backup engine transparently filters the blocks read from the tape. Users and applications are unaware that their backups are interleaved.

Parallel Backup Streams

In systems with high performance networks and volumes, it is possible to speed up large backup jobs by writing multiple tapes concurrently. This can be effective, for example, when full backups of large databases are made from storage snapshots. Each backup job processes one file at a time. If a database snapshot’s container files are divided into separate file lists, however, and the corresponding backup jobs are scheduled to occur concurrently, then several backup streams can be active at once, using different network links if they are available. Depending on the relative speeds of client, network, server, and tape drive, it may be appropriate to direct parallel jobs to different tape drives, or to multiplex them onto one tape.

V. Summing Up

Total Data Storage Management

This paper has discussed the four technology groups that comprise VERITAS *Total Data Storage Management* for e-business online data processing:

- a solid foundation consisting of flexible, failure tolerant volume management and an enterprise class file system,
- data migration and replication tools to get e-business online data to where it needs to be when it needs to be there,
- clustering technology to provide automated continuous application availability in the event of equipment, application, or site failures, and,
- backup and media management for distributed e-business requirements.

By employing these storage management technologies, e-businesses can ensure that their mission-critical data meets their performance, manageability, availability, and scalability requirements as they progress from start-ups to full-blown enterprises.