VERITAS Software Corporation

File System Performance
White Paper

VERITAS Software Corporation • 1600 Plymouth Street • Mountain View, CA 94043 • 1.800.258.8649 in the USA • 415.335.8000 • FAX 415.335.8050 • E-mail: vx-sales@veritas.com • World Wide Web: http://www.veritas.com/

VERITAS Software Corporation

File System Performance - White Paper

The VERITAS storage management product line has been designed in response to the needs of commercial computing environments. These are the systems that are now supporting mission critical applications and playing a major role in corporate computing services. Typical environments include online transaction processing systems, networked database servers, and high performance file services.

VERITAS specializes in on-line systems storage management technology which encompasses a product line that offers high performance, availability, data integrity, and integrated, online administration. VERITAS provides three complementary products: VERITAS FirstWatch®, VERITAS Volume Manager™, and the VERITAS File System™.

The VERITAS File System is a high availability, high performance, commercial grade file system providing features such as transaction based journaling, fast recovery, extent-based allocation, and online administrative operations such as backup, resizing, and defragmentation of the file system.

This is the second part in a series of VERITAS White Papers that discuss computer file systems. This second paper provides a discussion of the internals and mechanics of file system performance. General file system performance bottlenecks as well as some industry improvements will be discussed.

In this paper we will use a general UNIX file system model for discussion and introduce some general file system performance improvement techniques that exist both in the UNIX marketplace as well as across the entire file system industry. Where applicable we will discuss certain features of the VERITAS File System as they apply to file system performance.

Note: The audience for this paper includes systems engineers, administrators and integrators desire a breakdown and explanation of the general internals and mechanics of computer file system performance. For further reading on this subject, consult other VERITAS White Papers, including the *File System* and *VERITAS File System Performance* White Papers. These papers discuss general file system mechanics as well as the performance related internals, mechanics, benchmark test results and analysis, of the VERITAS File System.

A large part of the computer system's overall value can be defined as the information that resides in the file system. Some information system managers would contend that the **majority** of their system value is stored in their file system.

With this literal wealth residing on some type of electronic media there are several identified needs for information system managers as it applies to their computer's file system. First and foremost is integrity. An underwriting principle for the operation of any computer file system is that any information sent to the file system will be exactly the same when it is retrieved from the file system. A second identified need is availability. Since very few computers can operate without a file system, making sure that a computer file system is available, or at least not creating unavailability as a result of its own standard system operation, is a big concern.

Once you have established your file system's integrity and availability, the last major concern is performance. Making sure that the computer system can be accessed in a reasonable time, is a constantly moving target. To complicate matters, as computing technology advances, the amount of information to store has increased dramatically. All of which tends to complicate the issues of performance, availability, and integrity.

Note: It is evident that the cost of managing data and storage continues to rise. According to Strategic Research Corporation, disk storage requirements are growing at rates exceeding 60%, and the average yearly cost for managing storage is $350,000 for a PC LAN network and $750,000 for a UNIX network. In the March 1995 Strategic Research Corporation report entitled "Managing Network Storage", studies show that for every dollar spent on storage, a company will spend nearly $7 per year managing that same storage.

File System Performance - The Business Case

Before we go further it may be important to frame the discussion of file system performance relative to cost effectiveness. While agreement can easily be reached inside a company on the subject of computer file system integrity, this consensus typically starts to break down when considering the costs of availability. People have different requirements for how much of the system should be safeguarded against non-availability. Should uninterruptible power supplies be deployed to guard against power failures? If so on which machines?

These opinions further tend to divide when the issue of performance is brought up. The discussion revolves around how fast does the computer's performance need to be. More and more companies are putting more data online in their computer infrastructures. This presents the problem of making system performance a constantly shifting target. Does your performance need to equal performance of a year ago, even though you have added 50% more data? Does the system performance need to be better than your competitors? Does performance need to be different in systems available outside your company, as in the Internet, as opposed to inside, in the case of Intranets?

File System Performance and Availability

While system integrity and availability are requirements for most IS managers, performance is sometimes described as useful, but not a requirement. Part of this problem is defining the cost effectiveness of system performance. Does increasing system performance offer a return value for the investment? There certainly seems to be evidence that there is, if you look closer at the issue of system availability.

System performance shares a unique relationship with system availability. As a system begins to perform slower, users on that system must take longer time to do the same amount of work. This represents a dollar amount lost, directly related to computer system performance. Conversely if

computer system performance increases actually translate into increased worker productivity, there is a direct economic return on this system performance investment.

It is important to understand what types of system performance increases actually translate into increased productivity. While many IS managers have calculated a dollar amount for every hour of system downtime that is lost to their company, many have not analyzed these very same system downtime costs as they are related to decreases in system performance.

Finally once there is consensus on the need to improve the overall performance of your information system, the information system managers must allocate dollars to improve performance at various levels throughout the computer system. By far the most popular system improvements today are concentrated in the network communication channel. This not only indicates a popularity in computer network communications, but it also indicates the state of network communication technology. Next in no particular order comes the overall computing processor power, internal computer input / output (I/O) channels, overall system memory, and the computer's file system.

While there exist many opinions as to how these performance dollars should be allocated, it is important to realize that overall system performance begins with your computer's file system. This is where all information is stored to, and retrieved from, and the file system represents a large piece of your company's net worth. Upgraded, improved and tuned for performance properly, and your file system can increase performance throughout your computer information system. Relegated to the bottom of the system upgrade dollar pool and neglected, and it can literally sap the perceived value of your performance dollars, when they are allocated elsewhere in your system.

File System Performance Definition

Some file system performance discussions state that the type of work done by a computer affects file system performance. Defining what type of computing workload exists, helps to define what areas of file system performance are the most important.

If you operate your computer system in a manner that sequentially accesses files, a single file at a time, then you would need to be concerned with the amount of time it takes a single process, to satisfy a single disk I/O request. This is sometimes referred to as per-process disk throughput and is described as a synchronous environment since disk requests are handled mostly one at a time.

If your computer workload runs multiple processes and requests various files from random locations on the disk, then you would be concerned with your overall, or aggregate disk throughput. This type of environment can be described as a more concurrent environment. This is typical of many large server environments.

Both of these environments have some things in common. Systems that provide good aggregate disk throughput, oftentimes have good per-process disk throughput as well. However the system demands of a single threaded sequential access workload, differ from those of a multiple threaded random access workload. In some cases you could optimize your system to provide the best per-process throughput and then once your system was in place, with multiple users randomly accessing the disks, you could experience an actual performance degradation.

File Systems and Databases

As applications increase in size and type of workloads, some application vendors find that the standard file system design creates performance bottlenecks for their specific application workload. Database vendors in particular realize that due to the nature of their application workload, a standard file system, designed for general purpose workloads, actually introduces performance bottlenecks, simply because of the design.

As an answer to this, UNIX provides a method to completely remove the file system. This is commonly referred to as *raw I/O mode* or *raw I/O*. This mode removes the file system completely from between the application and storage devices. As a result, the application vendor must provide their

own file system services within their application. Since performance is of critical importance to database servers, many installations have taken to using raw I/O for any database server installations.

The problems that are introduced by this are not as visible in smaller installations. Since the application manages all file system services, then any file system services such as backup, administration, and restoring, must be done within the application. Most of these instances treat the raw system as one large image rather than separate files. System backups and restores must be done as a whole image, and performing maintenance on any one section of the raw system, can be very time consuming.

As database servers grow in size the management problems associated with these proprietary file system services, has magnified the need for better performing, standards based, file systems.

The type of workloads imposed on computers has changed dramatically over the last few generations of technology. In enterprise server installations it is difficult to find any instances of single threaded sequential access workloads any more. While the computer hardware industry has kept pace with many of these changes, the computer software industry has proceeded more slowly.

Many of the problems with technology improvements have to do with efforts to maintain backward compatibility. Particularly stung by this is the computer operating system software industry. A smaller group within this industry has been responsible for implementing advancements in computer file systems. However, due to its dependence on operating system vendors computer file system software technology has in some cases proceeded at an even slower pace. This has created an opportunity for third party company initiatives in the field of online storage management. In the open server commercial marketplace, one vendor who has responded is VERITAS Software.

The VERITAS Product Line

The VERITAS storage management product line has been designed in response to the needs of commercial computing environments. These are the systems that are now supporting mission critical applications, and playing a major role in corporate computing services. Typical environments include online transaction processing systems, both inter as well as intra-networked database servers, and high performance file services.

VERITAS specializes in systems storage management technology which encompasses a product line that offers high performance, availability, data integrity, and integrated, on-line administration. VERITAS provides three complementary products: VERITAS FirstWatch, VERITAS Volume Manager, and the VERITAS File System.

VERITAS FirstWatch is a system and application failure monitoring and management system that provides high-availability for mission-critical applications. FirstWatch dramatically increases server and application availability through the use of duplicate cluster monitoring processes on each node in a FirstWatch system pair. These monitoring processes communicate with each other over dedicated, duplicated heartbeat links.

The VERITAS Volume Manager is a virtual disk management system providing features such as mirroring, striping, disk spanning, hot relocation and I/O analysis. The VERITAS Visual Administrator™ exists as a graphical interface to VERITAS Volume Manager offering visual representation and management of the virtual disk subsystem including drag and drop features and simple or complex device creation.

The VERITAS File System is a high availability, high performance, commercial grade file system providing features such as transaction based journaling, fast recovery, extent-based allocation, and on-line administrative operations such as backup, resizing and defragmentation of the file system.

Computer system file performance is a constantly moving target. One contributing factor is that computer system technology improvements are not distributed equally among the component technologies. While the 1980s brought rapid improvements in processor and memory technology, sometimes increasing throughput and capacity by several orders of magnitude, the improvements in file system components improved at a much smaller rate. Typically magnetic disks during the same time period saw an increase in capacity by a factor of 10 and saw an increase in performance by only a factor of 2. This creates a situation where applications become *I/O bound*, and as a result are unable to take advantage of all of the improvements in processor and memory technology.

In this section we will define the file system layers from a performance point of view. These file system layers are defined here only for the purposes of this tutorial discussion. Other publications may define the file system layers differently, depending upon the specific discussion.

Our intent in this discussion is to point out the various file system components that can develop as file system performance bottlenecks. Once understood it becomes a simpler task to provide overall file system performance improvement. Additionally as we discuss the VERITAS File System it will become clear the performance improvements inherent in the design of the product.

Note: An important concept to remember regarding file system performance is that performance improvements must be balanced throughout the system. Increasing throughput on one level can simply lead to creating a new bottleneck at another level.

Generally speaking the layers involved in file system performance issues are:

- Software
  - Operating System
  - File System
  - Storage Hardware Device Driver
- Hardware
  - CPU / RAM
  - System Bus
  - Physical Disk Controller and Controller Bus
  - Physical Disk and Disk Bus

Improving performance in any one layer can sometimes increase performance overall. However it is very important to balance performance improvements throughout the file system so that increasing the performance in one component does not lead to performance bottlenecks in another component.

Some file system components' performance can be easily improved. One example is the physical disk, improved by installing faster disks. However, other areas are more difficult to change, such as the OS which would typically be updated only by the OS vendor. Perhaps by necessity, some of the slowest parts of the file system are ones that can be updated easily. This is true of the physical disk component. In fact, since by its very nature the physical disk is the most mechanical part of the file system, it also by far the slowest.

Note: Much of the performance improvements in all layers of the file system, are focused on the fact that the physical disk is the slowest link in the chain.

Understanding file system performance can generally be guided by the following points:

1) The physical disk is the slowest part of the storage system.

2) System memory is the fastest part of the storage system.

3) Delaying physical disk operations by temporarily storing file I/O in system memory, will improve performance. The caveat here is that the file system writes are reported completed to the system before they are actually committed to disk.

4) Organizing file system reads so that more information is retrieved than requested, will improve performance. The caveat here is that the extra information must eventually be requested.

5) When performing file I/O to the disk, organizing the read and write operations sequentially, based on the physical disk and not necessarily on the order the requests were received, will improve performance.

Note: The most efficient way to perform file system I/O is to organize all other file system operations so that I/O operations to the disk are performed in the longest sequential, or contiguous, manner possible.

File System Performance Technology

Some of the more common file system performance improvements have come at the physical disk level, with the improvement of physical disk speed. Other improvements have come from various vendors who have sought to increase performance at other layers in the file system. This includes the operating system and file system software layers. This next section will cover the basic performance improvements in operating system, and file system software layers.

Allocation Improvements

One of the techniques introduced to improve file system performance has to do with the way in which the disk space is allocated. File system performance enhancements here focus on allocation policies that basically improve on one thing. That is disk seek time. The premise is that if the system can allocate disk space in a more contiguous manner, then the time required for the disk heads to retrieve information from any one file will be greatly reduced. For the UNIX operating system, this design improvement involves making an indexed allocation system operate more like a contiguous allocation system.

Data Grouping

The UNIX operating system was improved over the original file system design by incorporating a technology known as data grouping. Since there exist two basic areas of the disk, the index information blocks and the file space blocks, the first design of the UNIX OS dedicated space on the beginning part of the disk to the index information, and left the remainder of the disk for the file space. The problem that resulted from this type of placement was that the larger the disk size became, the farther and farther the disk heads had to travel in order to read and update the index information.

Data grouping was introduced in later versions of the UNIX OS to alleviate this problem. The goal was to distribute the index information blocks throughout the file space, so that when a file was being retrieved, the disk head did not need to move very far to retrieve or update that file's index information. The actual design incorporates grouping sections of the disk blocks together and assigning some of the blocks to index information, and assigning the remaining blocks to file space for those index blocks. In most UNIX implementations these groups are referred to as cylinder groups.

Note: Sun Microsystems' UNIX implementation, called Solaris™, uses a data grouping model of *cylinder groups*. These cylinder groups are typically 4mb in size, and include inode information at the beginning of the cylinder group, followed by data blocks.

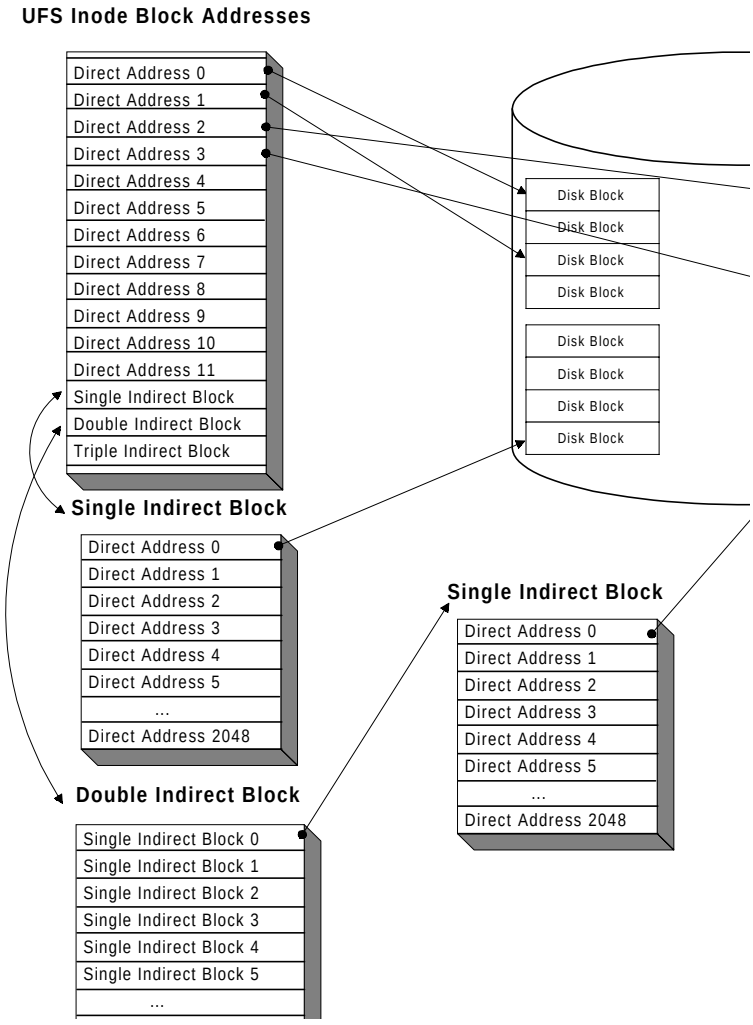
The VERITAS File System uses a data grouping model comprised of *allocation units*, or AUs. An allocation unit is typically 32mb in size, and includes some header information at the beginning of the AU, followed by data blocks. In the VERITAS File System, inodes are dynamically allocated, and stored in data blocks. This distributes the metadata workload more efficiently than UFS's cylinder group model.

UNIX File System Clustering

Sun improved on the general UNIX model of indexed allocation by introducing a technique called file system clustering in 1991. The basic change was to introduce a combination contiguous / indexed allocation method to improve file system performance, and to decrease the amount of CPU resources required for file system I/O. This effort was an attempt to provide more extent based performance for the UFS block based file system.

The basic change involved clustering disk blocks for allocation. The default cluster size is 56K. This means that when files are written, the writes themselves are buffered in system memory to see if they can collect at least 56K worth of writes before committing the write to disk creating a more contiguous allocation. This means that large chunks of a file are clustered together on sequential disk blocks (clusters) reducing the amount of time needed for the physical disk to retrieve them. File system reads are also performed in these larger cluster size chunks where possible.

This system operates much more like a contiguous allocation system for files smaller than the actual cluster size. When files need to be larger the file system allocates disk space in an indexed manner, albeit still trying to cluster the reads and writes. In addition to increasing the performance of the file system, Sun's testing at the time showed that by using file system clustering they were able to reduce the CPU workload of the file system by 25%.

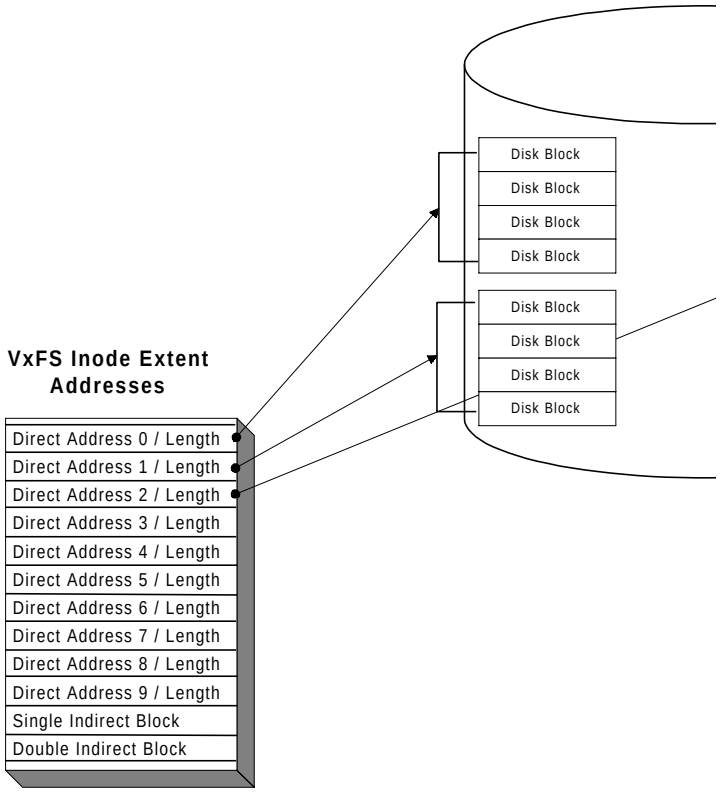

UFS Inode Block Addresses
Direct Address 0
Direct Address 1
Direct Address 2
Direct Address 3
Direct Address 4
Direct Address 5
Direct Address 6
Direct Address 7
Direct Address 8
Direct Address 9
Direct Address 10
Direct Address 11
Single Indirect Block
Double Indirect Block
Triple Indirect Block
Disk Block
Disk Block
Disk Block
Disk Block
Disk Block
Disk Block
Disk Block
Disk Block
Single Indirect Block
Direct Address 0
Direct Address 1
Direct Address 2
Direct Address 3
Direct Address 4
Direct Address 5
...
Direct Address 2048
Single Indirect Block
Direct Address 0
Direct Address 1
Direct Address 2
Direct Address 3
Direct Address 4
Direct Address 5
...
Direct Address 2048
Double Indirect Block
Single Indirect Block 0
Single Indirect Block 1
Single Indirect Block 2
Single Indirect Block 3
Single Indirect Block 4
Single Indirect Block 5
...

VERITAS Extent Based Allocation

Unlike traditional UNIX file systems, which assign space to files one block at a time, the VERITAS File System allocates blocks in contiguous segments called extents. Extent sizes are chosen based on the I/O pattern of the file, or may be explicitly selected to suit the application. Extent-based allocation can accelerate sequential I/O by reducing seek and rotation time requirements for access, and by enabling drivers to pass larger requests to disks.

Because a single pointer addresses more than one block, an extent-based file system requires fewer pointers and less indirection to access data in large files. UFS, with its 12 direct pointers, can only directly address up to 96KB of data (using 8KB blocks) without requiring at least one extra block of pointers and an indirect access. The VERITAS File System, with its 10 pointers to extents of arbitrary size, can address files of any supported size directly and efficiently.

When normal UNIX UFS needs to locate blocks in a file, it must perform steps to figure out which physical disk block correlates to the file block. This process involves following the UFS inode linking to locate the address of the correct physical disk block. This process is lengthened when larger files are involved due to the fact that larger files require double indirect block addressing at some point.



The VERITAS File System uses the much more efficient extent based allocation that dramatically improves the way in which large files are handled. Rather than linking indirect blocks of addresses, the VERITAS File System uses extent addresses which list a starting block address and a size. The disk blocks allocated for a file are stored in contiguous extents starting at the starting block address, and extending contiguously the number of blocks denoted by the size number.

What this translates to, is that when a large file is accessed in the VERITAS File System, the blocks needed can usually be found with no indirection, or directly. This direct addressing ability of the VERITAS File System dramatically increases the performance when the file system handles large files.

The VERITAS File System also provides interfaces for explicitly managing the layout of extents. Using a programming interface or a command, one can choose a fixed extent size for a file, require that its blocks all be stored contiguously, and reserve space for its future growth. This allows for optimal performance for applications which manage large files, such as voice and image data.

Disk I/O Cache

The fact that computer technology has improved on an unequal basis has lead researchers to determine that one viable method to improve overall file system performance is to decouple the CPU and disk performance. One of the most important decoupling techniques is *caching*. Caching is described as the ability of the system to store disk reads and writes in much faster main memory, rather than waiting to perform them through the disk channel.

Caching disk I/O in system memory is an effective way to improve the overall performance of the file system. This area of system memory is sometimes referred to as the buffer cache. Storing both reads and writes in the buffer cache allow three things to occur. The first is that the system can now store additional file blocks other than those requested, in system memory. Then once a read request is made for a block, there is the chance that it will already be in the much faster system memory, rather than needing to be retrieved from disk.

The second thing that can occur involves system writes. Storing writes temporarily in system memory allows an application to return to the user, stating that the write was done without actually having waited for the disk subsystem to perform the write.

The third thing that can occur by using the buffer cache to store disk I/O, is in the case of ordering disk activity. If you can take the list of I/O requests made to the disk subsystem and order them in a manner that will utilize the minimum amount of disk head movements, then overall file system performance will improve. Disk scheduling done at the storage device driver level addresses this area.

Read-Ahead Caching

Read-Ahead caching is something that is done, in terms of file system layers, by the type dependent file system. The type dependent file system basically performs calculations to determine the pattern of the disk requests being made. In some implementations, if sequential blocks of a file are being read from the disk, the type dependent file system will request additional file blocks in sequential order ahead of those requested. The intent is that when the request for the next file block is made, that the block in question is accessed from the buffer cache in memory, and not from disk. This is commonly referred to as read ahead caching.

Write-Behind Caching

Write-Behind caching is something that is again done by the type dependent file system. The type dependent file system stores all write requests in the buffer cache and then returns to the calling application that the write completed. This allows the storage device driver to schedule disk requests for all the pending disk writes in one group. A problem with this technique is that if the computer system were to go down prior to the write requests actually being written to the disk, the file system would have to be completely checked once it came back online, to insure its integrity. Any disk allocations that had not been completed would be discarded. To work around the need for this, most OSes allow a calling application to force a write to actually be completed on disk before it returns the completion.

Disk Scheduling

Another layer of the file system critical to overall file system performance is the storage device driver. One of the biggest jobs that the storage device driver accomplishes is the scheduling of disk requests. Disk scheduling orders I/O requests to the disk in such a manner as to optimize the physical movements of the disk heads. Some popular disk scheduling algorithms include First-come First-Served (FCFS) scheduling, Shortest seek time first (SSTF) scheduling, SCAN scheduling, sometimes called an elevator algorithm, and C-SCAN scheduling. It is oftentimes up to the storage system manufacturer as to which kind of disk scheduling to implement for their hardware.

VERITAS File System Journaling Performance

As described previously, file system journaling is a technique which involves committing system writes to a sequential log file. The benefit is that the writes are stored on disk, not in system memory, and the sequential nature in which they are written speeds up disk write activity.

The VERITAS File System employs a variation on the general logging technique by employing a circular *intent log*. All file system structure changes, or metadata changes, are written to this intent log in a synchronous manner. The file system will then periodically flush these changes out to their actual disk blocks. This increases performance by allowing all metadata writes to be written out to the permanent disk blocks, in an ordered manner out of the intent log.

Because the journal is written synchronously, it may also be used to accelerate small (less than or equal to 8KB) synchronous write requests, such as those used for database logging. Writes of this class may be written to the journal, a localized sequential block store, before they are moved to their places in the larger file system; this can reduce head movement and decrease the latency of database writes.

UNIX Raw I/O

There are three basic kinds of I/O in the UNIX OS. *Block devices*, like disks and tapes, *character devices*, like terminals and printers, and the *socket* interface used for network I/O. All of these I/O devices are insulated from the OS by device drivers. While character devices deal in streams of data traveling between applications and devices, block devices are noted for the fact that data travels between applications and devices in blocks. These block transfers are almost always buffered in the system buffer cache.

Almost every block device supports a character interface, and these are typically called *raw device interfaces*, or *raw I/O*. The difference with this interface is that none of the I/O traveling to or from the device is buffered in the system buffer cache. A limitation with this type of I/O is that depending on the device driver, the information transferred must be made in a specific block size needed by the device driver and device.

As a result, UNIX supports a raw I/O mode for applications. This provides applications with two things. As stated, this bypasses the UNIX system buffer cache. Applications using raw I/O mode in place of the file system, generally supply their own buffer cache system. The second item this provides, is that the UNIX system does not provide any locking services. Using raw I/O mode the application must guarantee data integrity by implementing their own data locking.

This ability to bypass the UNIX system buffer cache allows applications to define and manage their own I/O buffer cache. Database applications benefit from this technology expressly for the reason that the standard UNIX system buffer cache policies operate in way that is inefficient for some database applications.

The standard UNIX system buffer cache policy is to remove pages from the buffer cache on a least recently used (LRU) algorithm. This type of cache management provides good general purpose

performance for a broad range of applications. However, after examining these policies, some database application vendors have found that the performance of the database will increase if the system buffer cache employs a most frequently used (MFU) caching policy.

Using raw I/O allows a database vendor to employ an I/O system that is optimized to provide the best performance for their application. The largest problem that using raw I/O creates is that raw I/O disks do not contain a file system. Therefore the data on the disks cannot be accessed using file system based tools, such as backup programs. The larger the database the more problems this can create, especially when a small portion of a large database becomes corrupted, and in some implementations, the entire database must be restored in order to bring the system back online.

VERITAS Direct I/O

VERITAS implemented their Direct I/O feature in their file system to provide a mechanism for bypassing the UNIX system buffer cache while retaining the on disk structure of a file system. The way in which Direct I/O works involves the way the system buffer cache is handled by the UNIX OS. In UNIX, once the type independent file system, or VFS, is handed a I/O request, the type dependent file system scans the system buffer cache, and verifies whether or not the requested block is in memory. If it is not in memory the type dependent file system manages the I/O processes that eventually puts the requested block into the cache.

Since it is the type dependent file system that manages this process, the VERITAS File System uses this to bypass the UNIX system buffer cache. Once the VERITAS File System returns with the requested block, instead of copying the contents to a system buffer page, it instead copies the block into the application's buffer space. Thereby reducing the time and CPU workload imposed by the system buffer cache.

In order to ensure that Direct I/O mode is always enabled safely, all Direct I/O requests must meet certain alignment criteria. This criteria is usually determined by the disk device driver, the disk controller, and the system memory management hardware and software. First the file offset must be aligned on a sector boundary. Next the transfer size must be a multiple of the disk sector size. Finally depending on the underlying driver, the application buffer may need to be aligned on a sector or page boundary, and sub-page length requests should not cross page boundaries. Direct I/O requests which do not meet these page alignment requirements, or which might conflict with mapped I/O requests to the same file, are performed as data-synchronous I/O. This optimization, coupled with very large extents, allows file read accesses to operate at near raw I/O disk speed.

Another database application bottleneck with file systems, is that the UNIX OS maintains a single writer lock policy on each individual file block. This allows the OS to enforce a policy of system updates being guaranteed to be updated a single user at a time. This keeps file blocks from being corrupted via multiple simultaneous writes. However database applications lock data updates at a much finer level. Sometimes going so far as to lock updates based upon fields in a database record. As a result, locking an entire file block for data contained in a single field slows down database updates. Bypassing the file system and using a raw I/O interface allows the database vendor to lock system writes in a manner most efficient for their application.

VERITAS Quick I/O Database Accelerator

While the VERITAS File System implementation of Direct I/O improves the performance of things such as database application reads, the single writer lock policy of the UNIX OS continues to create performance bottlenecks for database system writes. In the VERITAS ServerSuite Database Edition 1.0, an additional feature has been added that bypasses the single writer lock policy in the UNIX OS. This feature is called the Quick I/O Database Accelerator. It allows a database application to lock data at the levels desired by the application, while bypassing the single writer lock for any given data block. This optimization allows file read and write accesses to operate at near raw I/O disk speed.

Disk Performance Technology

The physical disk itself, being the most mechanical, is almost always the slowest part of the system. In response, there have been many advancements in disk performance. The disk industry has also improved disk technology by advancing higher performance disk controller standards. One of the most popular current standards is SCSI which stands for Small Computer Standard Interface. The SCSI specification has gone through several revisions focused on increasing performance and reliability.

Physical Disk

Before discussing physical disk performance, we should define what makes up physical disk speed. In terms of measurement, disk speed is made up of approximately three parts. To access a particular block, the drive must move the head to the appropriate track or cylinder, this is called the *seek time*. Once the head is located at the right track, it must wait until the spinning platter, moves the correct block under the head, this is called the *latency time*. Lastly, the actual transfer of data between the disk and the system memory is called the *transfer time*. The total time required for the physical disk to service a request is the sum of the seek, latency, and transfer times.

The mechanical improvements in physical disk technology cover a wide range of topics. As the industry has matured, the disk media technology has undergone advancements. Disks which are collections of spinning platters with read/write heads for each platter, have gone from increasing the amount of platters, to increasing the density at which information is stored on the media, to increasing the speed at which the platters spin, in order to improve overall disk performance.

Other performance improvements employed at the disk level are the SCSI enhancements of *track buffering*, and *tag command queuing*, or TCQ. Track buffering is the controller's method of read-ahead caching. If a block in a sector is requested from the disk, the SCSI controller will read the entire sector into the controller's RAM, thereby decreasing the response time for any further requests from that sector. Tag command queuing (TCQ) is the SCSI drive's method for disk scheduling. Using advanced sensors on the drive to determine where the disk heads are located, TCQ will reorder the requests to optimize the drive head seek time.

Redundant Array of Inexpensive Disks (RAID)

In order to implement commercial-grade storage performance and reliability akin to those of main-frame disk subsystems on commodity disks (such as SCSI and IDE drives) and system buses (such as ISA, EISA, PCI, and SBus), techniques have been developed to combine multiple disks into single storage objects. These techniques are used to build Redundant Arrays of Independent Disks, or RAIDs. In papers written by Patterson et al. at the University of California at Berkeley, several RAID configurations (often called RAID levels) were proposed. In addition to RAID levels 2 through 5, which use parity calculations to provide redundancy, two other disk organizations were retroactively labeled as RAID: striping, or interleaving data across disks with no added redundancy, was identified as RAID level 0, and mirroring, maintaining full redundant data copies, was identified as RAID level 1. (RAID levels are often abbreviated as RAID-x; for example, RAID level 5 may be abbreviated as RAID-5.)

RAID-0 offers no increased reliability. However it can supply performance acceleration at no increased storage cost by sharing I/O accesses among multiple disks.

RAID-1 provides the highest performance for redundant storage, because it does not require read-modify-write cycles to update data (as does RAID-5), and because multiple copies of data may be used to accelerate read-intensive applications. However, it requires at least double the disk capacity (and therefore, at least double the disk expenditures). It is most advantageous for high-performance and write-intensive applications. Also, since more than two copies may be used, RAID-1 arrays may be constructed to withstand loss of multiple disks without interruption.

Use of mirroring (or RAID level 1) increases data availability and read I/O performance, at the cost of

sufficient storage capacity for fully redundant copies. RAID levels 2 through 5 address data redundancy by storing a calculated value (commonly called parity) which can be used to reconstruct data after a drive failure or system failure, and to continue to service I/O requests for the failed drive.

In order to increase reliability while preserving the performance benefits of striping, it is possible to configure objects which are both striped and mirrored. While not explicitly numbered in the RAID papers, this is sometimes called RAID-1+0, RAID-0+1, or RAID-10. This is done by striping several disks together, then mirroring the striped sets to each other, producing mirrored stripes. When striped objects are mirrored together, each striped object is viewed as if it were a single disk. If a disk becomes unavailable due to error, that entire striped object is disabled. A subsequent failure on the surviving copy would make all data unavailable. It is, however, extremely rare that this would occur before the disk could be serviced. In addition, use of hot spares makes this even less likely.

Among the parity RAID layouts, RAID-2 uses a complex Hamming code calculation for parity, and is not found in commercial implementations. RAID levels 3, 4 and 5 are often implemented. Each uses an exclusive-or (XOR) calculation to check and correct missing data. RAID-3 distributes bytes across multiple disks, requiring all I/O operations to access all disks; this accelerates single stream bandwidth-oriented applications, such as video servers, but does not perform well with applications such as databases that tend to read and write smaller blocks than one might effectively spread over multiple drives.

RAID-4 and RAID-5 compute parity on an application-specific block size, called an interleave or stripe unit, which is a fixed-size data region that is accessed contiguously. All stripe units at the same depth on each drive (called the altitude) are used to compute parity. This allows applications to be optimized to overlap read access by reading data off a single drive while other users access a different drive in the RAID. These types of parity striping require write operations to be combined with read and write operations for disks other than the ones actually being written, in order to update parity correctly. RAID-4 stores parity on a single disk in the array, while RAID-5 removes a possible bottleneck on the parity drive by rotating parity across all drives in the set.

RAID-5 allows redundancy with less total storage cost. The read-modify-write cycles it requires, however, will reduce total throughput in any but read-only or extremely read-intensive cases, and the loss of a single disk will cause read performance to be degraded, requiring the system to read all other disks in the array and recompute the missing data. (This is true for both controller-based and host-based RAID-5.) In addition, it does not withstand the loss of multiple disks, and cannot be made multiply redundant.

While RAID level 2 is not commercially implemented, and RAID level 3 is likely to perform significantly better in a controller-based implementation, RAID levels 4 and 5 are more amenable to host-based software implementation. RAID-5, which balances the actual data and parity across columns, is likely to have fewer performance bottlenecks than RAID-4, which will require access of the dedicated parity column for all read-modify-write accesses.

VERITAS Volume Manager RAID Support

VERITAS Volume Manager is a disk management product that augments the UNIX disk partitioning model. It extends the operating system to transcend limits on disk capacity, performance, and reliability, incorporating RAID levels 0 (striping), 1 (mirroring), 1+0 (mirrored stripes), and 5 (distributed-parity striping). VERITAS Volume Manager provides on-line space allocation and configuration management, error handling, performance analysis, and operation tracing, allowing the administrator to ensure optimal use of storage resources. VERITAS Volume Manager may be used to enhance both file system services (including NFS™) and on-line DBMS engines (among them Oracle, Sybase®, and Informix®). It is integrated into the operating system as a set of loadable device drivers, a library and a utility set, and thus, does not require the replacement of any standard operating system components.

Mirroring or RAID-1, may be used to accelerate multi-user read-oriented applications. Multiple read requests may be serviced from separate instances of the data, allowing the I/O load to be distributed across multiple spindles, and potentially reducing the effect of head movement on data access as well.

VERITAS Volume Manager supports two read policies. One, called the round-robin policy, satisfies reads from alternate copies of the data, as described above. It is most applicable when multiple copies of the data are stored on devices of similar latency and transfer capabilities. This read policy has been enhanced to alternate disks only if the current read operation is greater than 64k bytes distant from the previous operation; this allows benefit from disk and controller read-ahead caches, as well as from helping to minimize head movement.

The second policy, the preferred-plex policy, may be used when one instance of the data is on a device that is significantly faster than the other copies. This may be a disk with different characteristics, a spanned object that is mirrored to a non-spanned object, or a different fundamental technology, such as a RAM disk or solid-state disk. In this case, it is advantageous to service all reads from that faster copy (while writing all data to all copies).

By default VERITAS Volume Manager will attempt to select a read policy based on the current volume configuration. If all data copies are of the same basic layout (all striped or all non-striped), VERITAS Volume Manager will assume that round-robin is optimal. If some are striped and others non-striped, it will assume that one of the striped instances is to be used with a preferred-plex policy. VERITAS Volume Manager performs all writes in parallel; this causes total elapsed time for mirrored writes to be, in general, minimally greater than the elapsed time for a single write. VERITAS Volume Manager supports up to 32 instances of the data in a volume. These plexes, or mirror instances, are used to increase data reliability, to accelerate access for read-intensive applications, and to support on-line administrative operations (described below).

VERITAS Volume Manager allows partial, or sparse, instances of the data in a volume. This may be useful where some of the data is read much more frequently than the rest; mirroring these “hot spots” to a faster but more expensive medium (RAM disk, solid-state disk, or simply a faster device) and marking that plex as the preferred copy for reads may result in significant acceleration for read-intensive applications.

File System Alignment

While RAID technology increases performance in some implementations, tuning RAID systems for proper file system alignment can increase performance for most striped RAID configurations. In most commercial implementations this involves using RAID-1, RAID-5 and RAID-0+1 configurations.

The technique behind file system alignment involves setting the layout of the file system across the drive array in such a manner that the workload is distributed as equally as possible. In order to accomplish this there must be a determination as to what sections of the file system to distribute, and there must be a method for aligning these sections.

This can be accomplished with most modern file systems that use data grouping techniques. As mentioned previously, data grouping was introduced in later versions of the UNIX OS to alleviate the problem of disk seek time increases as the separate index and data sections of a file system were spaced farther apart on the disk. The actual design incorporates grouping sections of the disk blocks together and assigning some of the blocks to index information, and assigning the remaining blocks to file space for those index blocks. In most UNIX implementations these groups are referred to as cylinder groups.

The beginning of a UNIX UFS cylinder group contains the metadata blocks, and these blocks tend to be centers of disk activity. Aligning the UFS file system so that the cylinder groups begin on different drives in the array will align the file system for this method. Using this technique allows the separate drives in the array to perform the highest amount of simultaneous accesses.

The way in which this can be accomplished is by the setting of the RAID stripe unit size. This is the size of disk space, on each disk, that is accessed in one pass. The combined total stripe size of all the disks is known as the RAID stripe width. Setting the stripe size to 512KB on a 3 column (disk)

RAID-0 array, would result in a stripe width of 1.5MB (512x3).

Since the cylinder group size in UNIX is typically 4MB, setting the stripe unit size to 512K for a 3 column array as described, would mean that the beginning of each subsequent cylinder group begins on a different drive in the array.

The VERITAS File System's cylinder groups, called allocation units (AU), do not contain similar metadata blocks at the beginning of the AU. Inodes are allocated dynamically within the data blocks of each AU. What is more important in terms of file system alignment for this file system is keeping the AUs allocated on even disk boundaries. This provides increased performance throughout the file system, as well as allow Direct I/O technologies to be utilized. What this necessitates is padding the AUs so that they begin and end on even disk boundaries. In the 2.3 version of the VERITAS File System this is done automatically if the disks are being managed by the 2.3 version of the VERITAS Volume Manager.

When looking at overall system performance, the performance of the file system can have far reaching impact. Improving performance at the file system level, benefits every aspect of computer system performance. In a similar light, since performance improvements have increased at a faster pace in all other computer component technologies, improving the performance of a computer's file system will almost never create a bottleneck in other parts of the system.

We have illustrated that a computer file system is composed of multiple layers interacting with one another to perform computer file I/O. Each layer in a computer file system is responsible for a specific set of functions, and each layer typically operates by receiving requests from one specific layer, processing the request, and then handing the result to another specific layer. Some redundancy in this approach can lead to performance bottlenecks in the operation of file systems.

We have also illustrated a number of improvements on the performance of file system layers. It is important to remember that improving performance in one layer can sometimes lead to bottlenecks in other layers. When considering file system performance improvements, it is important to balance improvements throughout the file system. We have also discussed that some layers in a computer file system have been designed to be easily upgraded, as is the case with physical disks, compared to other layers in a computer file system, such as the type dependent file system.

Finally, while some areas of the computer file system industry have scaled well to increasing user demands, for example disk drive manufacturers have steadily increased the size of disk drives, other areas of the industry have not kept pace. Even with their advancements, operating system vendors have been slow to improve the scalability of their legacy file system software components with backwards compatibility being a large factor.

The VERITAS Software File System fills this gap to provide scaleable, commercial class, next generation file system technology.

For further file system technical information consult other VERITAS White Papers, such as the *File System* and *VERITAS File System Performance* white papers, available from VERITAS Software.