A Storage Networking Appliance

Dave Hitz and Akshay Bhargava, Network Appliance, Inc.
February 2006, TR-3001

Table of Contents

1. Introduction

An appliance is a device that performs a single function very well. A popular and accelerating trend in networking has been to use appliances instead of general-purpose computers to provide common services. For instance, special-purpose routers from companies like Cisco Systems and Nortel Networks have almost entirely replaced general-purpose computers for packet routing, even though general-purpose computers originally handled all routing. Similarly, modern printers are more likely to plug into the network than into a general-purpose computer. Other examples of network appliances include network terminal concentrators, network FAX servers, and network backup servers.

Appliances have been successful because they are easier to use, more reliable, and have better price/performance than general-purpose computers. These benefits arise because appliances can be optimized specifically for their single function, without the compromises necessary to meet the many conflicting requirements of a general-purpose system.

Network Appliance invented and pioneered the network storage appliance (formerly referred to as a filer) to bring the advantages of an appliance to the Windows® and UNIX® market. Storage appliances cannot run applications and do not run a general-purpose operating system like UNIX or Windows NT®. For NFS and CIFS file service its ease of use and price/performance cannot be matched.

2. Benefits of a Storage Appliance

Storage appliances have benefits similar to other network appliances: ease of administration, high performance with cost-effective hardware, high system and data availability, rapid system deployment and scalability, and low total cost of storage ownership.

2.1 Ease of Administration

Storage appliances are easy to administer because they eliminate operating system components and architecture unrelated to file service. Even options related to file service have been simplified or eliminated wherever possible. For instance, the storage appliance has file systems that grow automatically when new disks are added. This eliminates the complexity of partitioning disk drives and assigning the partitions to different users. Logical partitions can be expanded and shrunk dynamically with zero downtime.

A storage appliance simplifies backup with Snapshot™ copies, which are online, read-only copies of the entire file system. NetApp Snapshot technology is a feature of the WAFL® (Write Anywhere File Layout) storage virtualization technology that is a part of Data ONTAP®, the microkernel that ships with every NetApp storage system. A NetApp Snapshot copy is a "frozen," read-only view of a WAFL volume that provides easy access to old versions of all of the data contained within the WAFL volume, including files, directory hierarchies, and/or LUNs (logical unit numbers). Snapshot copies can be created automatically several times a day, and users can access Snapshot copies over NFS or CIFS to examine or recover old versions of their files without help from the system administrator. A storage appliance creates Snapshot copies using a highly optimized technique that consumes no disk space until files referenced by a Snapshot copy are deleted or modified.

Storage appliances offer multiple options to administer the system, although minimal administration is actually involved when compared to conventional storage subsystems. A Web-based graphical interface (FilerView®) provides the administrator with a simple-to-use mechanism to manage all aspects of a storage appliance. Windows administrators can utilize tools they are familiar with to manage tasks specific to Windows using the Microsoft® Management Console (MMC) for Computer Management and Performance Analysis. For command-line enthusiasts, a simplified command-line interface similar to UNIX is available that provides many familiar commands to any UNIX system administrator. These commands include file service and networking commands such as *ping*, *ifconfig*, *exportfs*, and *nfsstat*, as well as more general

commands such as *date*, *uptime* and, *passwd*. The command-line interface can be accessed either locally from the console or remotely via telnet.

2.2 Price/Performance

To obtain high performance at a reasonable cost, the storage appliance uses a very lightweight, real-time microkernel running on standard-based hardware. Performance is improved by eliminating superfluous features, such as virtual memory, graphical window systems, and local applications. Software optimization has proven to be a significantly more cost-effective approach for obtaining high performance levels versus resorting to complex and esoteric hardware architectures. Complex architectures increase expense and reduce reliability. Performance is excellent because storage appliance software (Data ONTAP) was designed and optimized from the ground up for network file service.

2.3 System and Data Availability

The appliance approach improves data availability. A general-purpose computer has so many different features and applications that it is impossible to test all possible usage patterns. An appliance can be tested much more thoroughly because it does just one thing. The simpler hardware that an appliance uses also reduces the number of points of failure.

Storage appliances use RAID (Redundant Array of Inexpensive Disks) to protect against data loss caused by disk failure. RAID also improves availability because a storage appliance can run even with failed disks. The storage appliance's ability to boot in just a minute or two further reduces downtime when a system failure occurs or when a system upgrade is being installed.

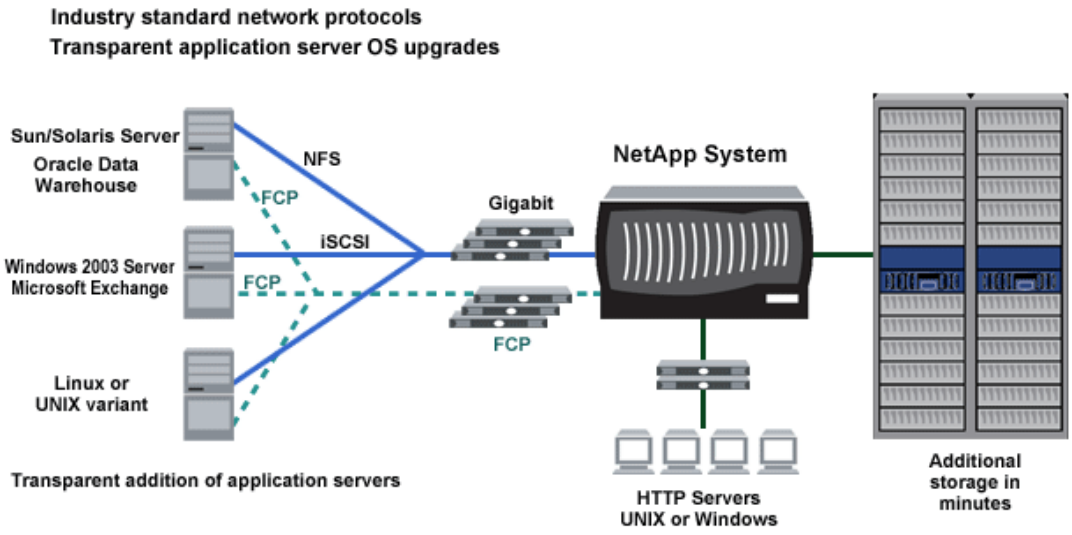
Data availability is improved by key software features within the storage appliance, including:

- Snapshot technology for instant file recovery
- SnapRestore® for instant file-system recovery
- SnapMirror® for replication for disaster recovery
- SnapVault® for data protection
- SnapManager® for fast recovery of Microsoft Exchange and SQL Server

The appliance approach also significantly minimizes planned downtime when compared to conventional storage subsystems.

2.4 Rapid Deployment and Scalability

A storage appliance installs in less than 30 minutes and is RAID-enabled from the start, so administrators need not spend hours setting up RAID groups and volumes. Storage can be added dynamically to any file system with a single command. Adding storage to storage appliances incurs no system downtime and does not impact performance or normal operation. Because storage appliances are network-attached, application servers can be dynamically added to the environment with zero downtime and significantly less effort than traditional SAN solutions.



2.5 Low Total Cost of Storage Ownership

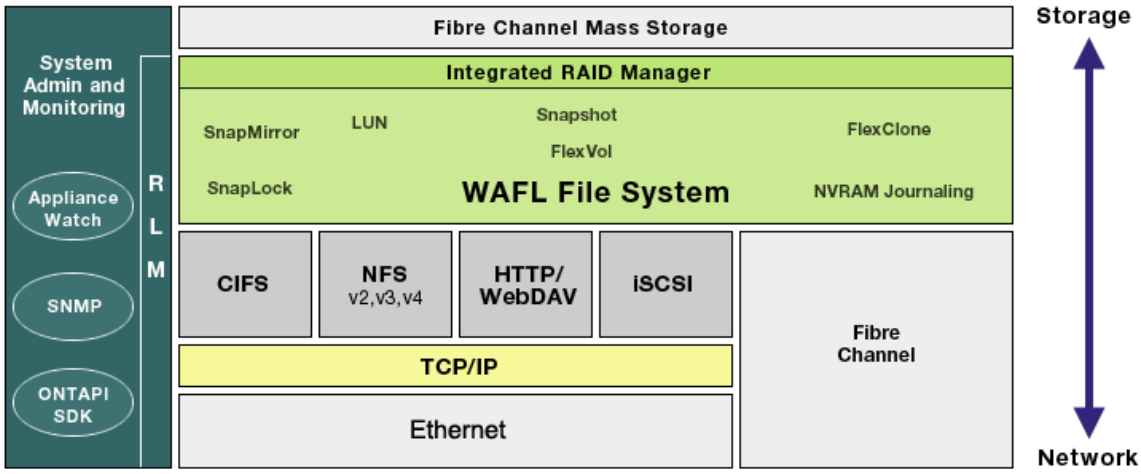
The appliance architecture results in low cost of storage ownership by offering both a low initial investment cost and, more important, significantly lower operational costs. Examples of lower operational costs include:

- 3-minute upgrades
- 1-command file system expansion
- Minimal RAID management
- Deleted files can be retrieved easily without resorting to tape (Snapshot copies)
- File systems can be instantaneously reverted to a previous point in time
- Unsuccessful third party software upgrades do not result in reinstalling the software
- Quick recovery from application corruption issues

3. Storage Appliance Architecture

The basic architecture of the storage appliance's software is shown in Figure 1. A collection of tightly coupled processing modules handle NFS, CIFS, and HTTP requests. A request starts in the network driver at the lower left, and moves up through network protocol layers and the file system, eventually generating disk I/O if necessary. When the file system finishes the request, it sends a reply back to the network. The administrative layer at the top supports a command-line interface similar to UNIX that monitors and controls the modules below. In addition to the modules shown, a simple real-time kernel provides basic services such as process creation, memory allocation, message passing, and interrupt handling.

A multitasking real-time kernel provides basic services for the whole system. A tightly coupled set of modules handles NFS, CIFS, and HTTP requests. The administrative layer at the left-hand side supports a command-line interface like that in UNIX that monitors and controls the modules below.



The networking layer is derived from the same Berkeley code used by most UNIX systems, with modifications to communicate efficiently with the storage appliance's file system. The storage appliance provides transport-independent, seamless data access using block- and file-level protocols from the same platform. The storage appliance provides block-level data access over a Fibre Channel SAN fabric using FCP and over an IP-based Ethernet network using iSCSI. File access protocols such as NFS, CIFS, HTTP, or FTP provide file-level access over an IP-based Ethernet network.

WAFL, the storage appliance's file system, was specifically designed to work in a network file server appliance. It is described in the next section, "WAFL: Write Anywhere File Layout." WAFL and RAID were designed together to avoid the performance problems that most file systems cause with RAID. RAID and its interaction with WAFL are described in more detail in the section "RAID: Redundant Array of Inexpensive Disks."

4. WAFL: Write Anywhere File Layout

Three requirements drove the decision to design a new file system for the storage appliance:

1. The file system should operate efficiently with RAID.
2. The file system should be able to grow dynamically as new disks are added.
3. The file system should not require any time-consuming consistency checks in the normal course of events.

The ability to support Snapshot copies emerged from these requirements.

In some respects, the WAFL disk format is similar to that of other UNIX file systems such as the Berkeley Fast File System and the IBM TransArc Episode file system. For example:

- WAFL is block based, using 4KB blocks with no fragments.
- WAFL uses inodes to describe its files.
- Directories are specially formatted files.

Like Episode, WAFL uses files to store meta-data. The three most important WAFL meta-data files are the inode file (which contains all inodes), a free block bitmap file, and a free block count file. Keeping meta-data in files allows meta-data blocks to be written anywhere on disk. This is the origin of the name WAFL, which stands for Write Anywhere File Layout. WAFL has complete flexibility in its write allocation policies because

no blocks are permanently assigned to fixed disk locations as they are in the Berkeley Fast File System (FFS).

WAFL uses this flexibility to optimize write performance for the storage appliance's RAID features. The importance of write-optimized file systems has been known for some time. In a UNIX or Windows file server, writes are especially important because they must be written to disk (or NVRAM), while reads are cached in memory at the UNIX or Windows client and the server. The result is that disks on UNIX and Windows servers commonly have several more times as many write operations as reads. The section "RAID: Redundant Array of Inexpensive Disks" describes the write allocation policies that WAFL uses in order to work efficiently with RAID4 and RAID-DP™.

Starting with Data ONTAP 7G, WAFL introduced a new building block known as flexible volumes. The FlexVol™ technology decouples the previous direct connection between volumes and their associated physical disks, vastly increasing flexibility and storage efficiency. An aggregate provides the connection between the logical flexible volume and the underlying physical storage and isolates the volume from this connection. A flexible volume is able to stripe all of its data across the entire aggregate.

FlexClone™, another new technology introduced with Data ONTAP 7G, makes it possible to replicate volumes instantaneously without duplicating shared blocks. Each FlexClone copy can itself be replicated, allowing many possibilities for branching of storage environments, similar to branching of source code trees within modern source code control systems. Virtualization, common to generations of application programmers when applied to memory allocation and management, is now available to data storage managers.

4.1 Snapshot Technology

Snapshot copies are another benefit of the WAFL write anywhere approach. A Snapshot copy is an online, read-only copy of the entire file system. Typically a Snapshot copy only takes a few seconds to create-usually less than one second, regardless of the size of the volume or the level of activity on the NetApp storage system. After a Snapshot copy has been created, changes to data objects are reflected in updates to the current version of the objects, as if Snapshot copies did not exist. Meanwhile, the Snapshot version of the data remains completely stable. A NetApp Snapshot copy incurs no performance overhead.

A Snapshot copy can be used as an online backup capability, allowing users to recover their own files. A Snapshot copy also simplifies backup to tape. Since a Snapshot copy is a read-only copy of the entire file system, it allows self-consistent backup from an active system. Instead of taking the system offline, the system administrator can make a backup to tape of a recently created Snapshot copy.

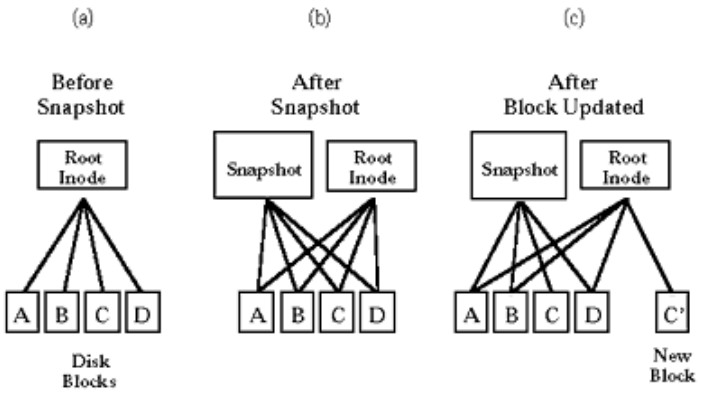


Figure 1: Snapshot copy creation

Figure 1 (a) is a simplified diagram of the file system, with a root structure at the top pointing to disk blocks. Figure 1 (b) shows a Snapshot copy being created by duplicating the root structure. Figure 1 (c) shows block C being updated, so that the file system points to new data in C' while the Snapshot copy still points to the original data in C.

Figure 1 shows how a Snapshot copy works. A WAFL file system can be thought of as a tree of blocks rooted by a data structure that describes the inode file. The inode file in turn contains the inodes that describe the rest of the files in the system, including meta-data files such as the free block bitmap and the free inode bitmap. Figure 1(a) is a simplified representation of a complete file system with the root data structure shown at the top.

WAFL creates a new Snapshot copy by making a duplicate copy of the root data structure, as shown in Figure 1(b). Since the root data structure is only 192 bytes, and since no other data blocks need to be copied on disk, a new Snapshot copy does not actually consume any additional disk space until a user deletes or modifies data in the active file system. WAFL creates a Snapshot copy in just a few seconds. Figure 1(c) shows what happens when a user modifies data block C. WAFL writes the new data to block C' on disk, and changes the root structure for the active file system to point to the new block. The Snapshot copy still references the original block C, which is unmodified on disk.

The storage appliance can keep up to 255 Snapshot copies per volume. The Snapshot copies can be created manually or the storage appliance can create and delete them automatically according to a user-defined schedule. How long a Snapshot copy is kept depends on how quickly the file system changes. In many environments, most data in the file system remains unchanged from day to day. A 300GB database may only change 50 to 100MB per day. When files change slowly, Snapshot copies can be kept online for many days, weeks, or months before they begin to consume unacceptable amounts of disk space. On the other hand, file systems in some environments change very quickly. Sites using applications such as CAD, which frequently overwrites many large files, may overwrite the entire contents of the file system in just a day or two. In such an environment there may not be room to keep Snapshot copies for more than a few hours, if at all.

4.2 User Access to Snapshot Copies

Every directory in the file system contains a special subdirectory that allows users to access the Snapshot copy, providing access to earlier-in-time versions of the file system. Suppose that a user has accidentally removed a file named foo and wants to recover it from a Snapshot copy. This example shows how to list all saved versions of foo from a UNIX or NFS client:

```
% ls -lu  .Snapshot/*/foo
-rw-r--r-- 1 hitz 16787 Jun 16 15:00  .Snapshot/hourly.0/foo
-rw-r--r-- 1 hitz 16744 Jun 16 12:00  .Snapshot/hourly.1/foo
-rw-r--r-- 1 hitz 16811 Jun 16 10:00  .Snapshot/hourly.2/foo
```

Three Snapshot copies contain foo. With the -u option, ls shows foo's access time which indicates when the Snapshot copy was created. To recover the most recent version, the user can simply copy that version of foo back into the current directory:

```
% cp .snapshot/hourly.0/foo .
```

Listing .snapshot/hourly.0 would show all of the files that the current directory contained when the hourly.0 Snapshot copy was created. The .snapshot directories are "hidden" in the sense that they do not show up in directory listings. If .snapshot were visible, commands like find would report many more files than expected, and commands like rm -rf would fail because Snapshot files are read-only and cannot be removed.

Windows users will see a ~snapshot folder as illustrated in the following figure.

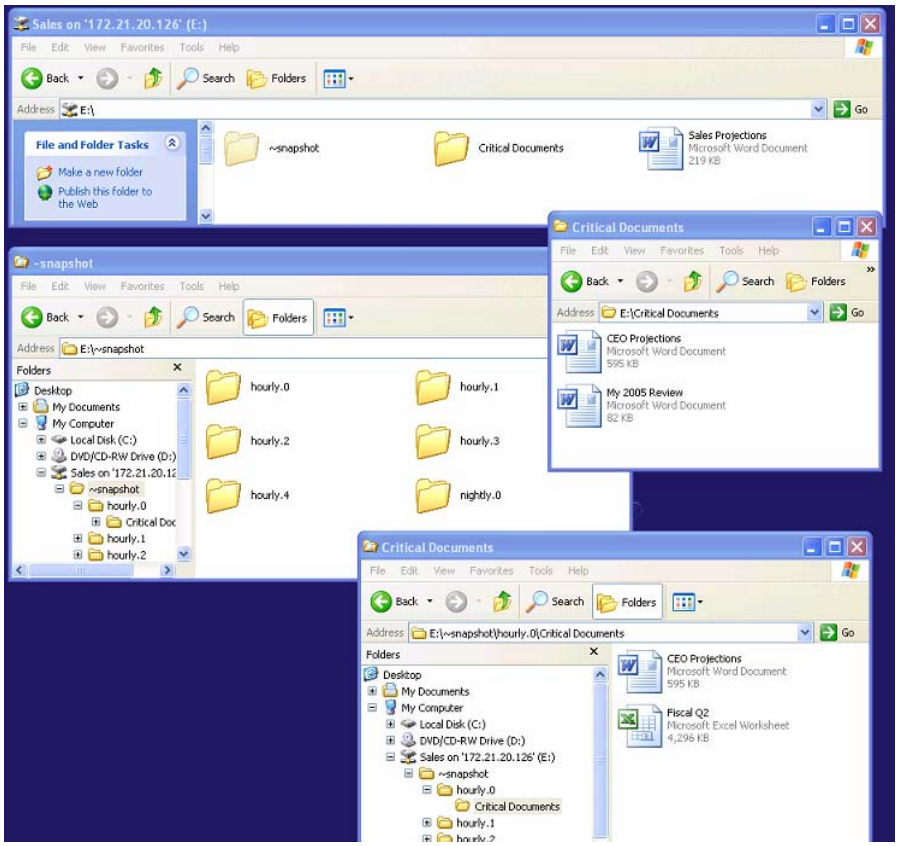


Figure 2

4.3 Consistency Points and NVRAM

At least once every 10 seconds WAFL generates an internal Snapshot copy called a consistency point so that the disks contain a completely self-consistent version of the file system. When the storage appliance boots, WAFL always uses the most recent consistency point on disk, which means that even after power loss or system failure there is no need for time consuming file system checks. The storage appliance boots in just a minute or two, most of which is spent spinning up disk drives and checking system memory.

The storage appliance uses battery-backed up non-volatile RAM (NVRAM) to avoid losing any data input/output requests that might have occurred after the most recent consistency point. During a normal system shutdown, the storage appliance turns off protocol services, flushes all cached operations to disk and turns off the NVRAM. When the storage appliance restarts after a system failure or power loss, it replays any protocol requests in the NVRAM that have not reached disk.

Using NVRAM to store a log of uncommitted requests is very different from using NVRAM as a disk cache, as some UNIX products do [Lyon89]. When NVRAM is used at the disk layer, it may contain data that is critical to file system consistency. If the NVRAM fails, the file system may become inconsistent in ways that fsck cannot correct.

WAFL uses NVRAM as a file system journal, not as a cache of disk blocks that need be changed on the drives. As such, WAFL use of NVRAM space is extremely efficient. For example, a request for a file system to create a file can be described in just a few hundred bytes of information, where as the actual operation of creating a file on disks might involve changing a dozen blocks of information or more. Because WAFL uses NVRAM as a journal of operations that need to be performed on the drives, rather than the result of the operations themselves, thousands of operations can be journaled in a typical storage appliance NVRAM log.

4.4 FlexVol and FlexClone

FlexVol and FlexClone technologies optimize storage utilization and simplify management.

FlexVol enables volumes to be treated as logical data containers that can be sized, resized, managed, and moved independently from the underlying physical storage. This enhances a storage administrator's ability to address a variety of data management requirements while preserving the familiar semantics of volumes and the current set of volume-specific data management and space allocation capabilities.

FlexClone technology enables system administrators to instantly create clones of flexible volumes. A FlexClone volume is a writeable point-in-time image of a FlexVol volume or another FlexClone volume. FlexClone technology adds a new level of agility and efficiency to storage operations.

FlexVol and FlexClone technology is discussed in more detail in "Introduction to Data ONTAP Release 7G" [TR3356].

5. RAID: Redundant Array of Inexpensive Disks

The storage appliance uses either RAID Level 4, which stores all parity information on a single disk, or RAID-DP, which stores all parity information on two disks to protect against disk failures. Unlike generic RAID4 or RAID5 implementations which are architected without thought to filesystem structure and activity, the WAFL RAID implementation is heavily optimized to work in tandem with the file system. By optimizing the file system and the RAID layer together, the NetApp RAID design provides all the benefits of parity RAID protection, without incurring the performance disadvantages associated with general-purpose RAID5 solutions. Also, because the WAFL RAID4 design does not interleave parity information like RAID5, the overall system can still be expanded quickly and easily, even though RAID protection is present.

With RAID4, if one block on a disk goes bad, the parity disk within that disk's RAID group is used to recalculate the data in that block, and the block is mapped to a new location on disk. If an entire disk fails, the parity disk prevents any data from being lost. When the failed disk is replaced, the parity disk is used to recalculate its contents automatically.

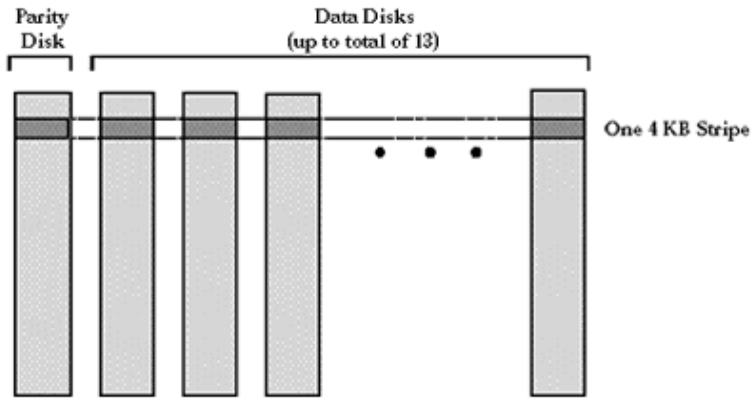


Figure 3: RAID4 Disk Layout

A RAID4 array uses one disk for parity and the rest for data. Each disk is made up of 4KB blocks. A stripe consists of one block from each data disk and one block from the parity disk. The parity block in each stripe allows data to be recalculated if any one block in the stripe is lost.

Figure 3 shows how the storage appliance's RAID4 disk array is divided into stripes, each one of which contains one 4 KB block on the parity disk along with one 4KB block from each of the data disks. It is convenient to think of a disk block consisting of a very large number of bits, 32,768 bits. For each bit position in the block, the bit on the parity disk is determined by the EXCLUSIVE-OR of the corresponding bits on the data disks.

This is easier to understand if you think of each 4 K-byte disk block as being a really big integer (32 K-bits long), and you think of RAID as doing simple math on these integers. The parity block is basically the sum of the blocks in the stripe.

```
Parity     Data 1     Data 2     Data 3
  12         3          7          2
```

If one of the data disks fails, for instance "Data 2," then the data can be reconstructed with simple arithmetic:

```
Data 2 = Parity - Data 1 - Data 3
       =   12   -  3     -  2
       =    7
```

In reality, the RAID system uses EXCLUSIVE-OR instead of addition and subtraction, and the numbers would be large, instead of the small integers shown here. But the math works out the same, and using addition and subtraction on small numbers makes the technique easier to understand.

Lost data is recalculated on the fly to let the system run even with a failed disk. The entire contents of a failed disk are recalculated after it has been replaced. Of course, if two blocks in a single stripe fail, there is no longer sufficient information to recalculate the lost data. If the parity disk itself fails, it must be replaced, but no file system data has been lost.

RAID-DP dramatically increases data fault tolerance from various disk failure scenarios. RAID-DP can recover from double disk-failure while allowing the RAID group to continue serving data and recreate data lost from up to two failed disks. RAID-DP is discussed in detail in "NetApp Data Protection: Double Parity RAID for Enhanced Data Protection with RAID-DP" [TR3298].

5.1 Other RAID Levels

The original description of RAID described RAID levels 1 through 5. The most commonly used RAID levels are 1, 3, and 5. RAID 1 is simply disk mirroring, in which all data is duplicated on two separate disks. RAID 1 is very safe, but it doubles the cost of disk storage.

RAID 3 is like RAID4, in that it uses a single parity disk, but stripes in RAID 3 are so small that each individual read or write operation must access all disks in the array. For instance, the first byte in a block of data might be on the first disk, the second byte on the second disk, and so on. RAID 3 systems often keep the disk heads synchronized to reduce latency. RAID 3 is a good fit for applications that require a very high data rate for a single large file, such as super-computing and graphics processing. It performs poorly with multi-user applications that generate many unrelated disk operations in parallel because each operation generates traffic on each disk in the array. By contrast, each data disk in a RAID4 array can satisfy a separate user request at the same time.

RAID5 is like RAID4, but instead of keeping all parity blocks on a single disk, it cycles parity among all of the disks in the array: parity for the first stripe is on the first disk, parity for the second stripe on the second disk, etc. The primary advantage of RAID5 is that it prevents the parity drive from becoming a bottleneck. (See "Eliminating the Parity Disk Bottleneck" below for how WAFL avoids this bottleneck for the storage appliance.) The primary disadvantage is that it is not practical to add a single disk to a RAID5 array because to add new disks easily, a new array must be added. Thus, if a RAID5 implementation uses 7 disks in each array, then disks must normally be added 7 at a time.

Some people have used the term RAID 0 to refer to disk striping, which is basically RAID4 without the parity disk. Since there is no redundancy in disk striping, applying the term RAID to it is misleading.

5.2 Eliminating the Parity Disk Bottleneck

Most vendors of RAID peripherals for UNIX and Windows have avoided RAID4 because with general-purpose file systems, the parity disk becomes a bottleneck. The WAFL file system, on the other hand, uses the flexibility of its "write anywhere" layout to write blocks to locations that are efficient for RAID4.

As an example of how WAFL differs from general-purpose file system designs, the FFS (the Berkeley Fast File System) was designed to optimize writes for one file at a time. As a result, it typically writes blocks for different files to widely separated locations on disk. Figure 4(a) shows how FFS might allocate blocks for 3 unrelated files in a RAID array. While each data disk in the example has only 2 writes, the parity disk has 6. More importantly, the parity writes are widely spread, causing time consuming seeks.

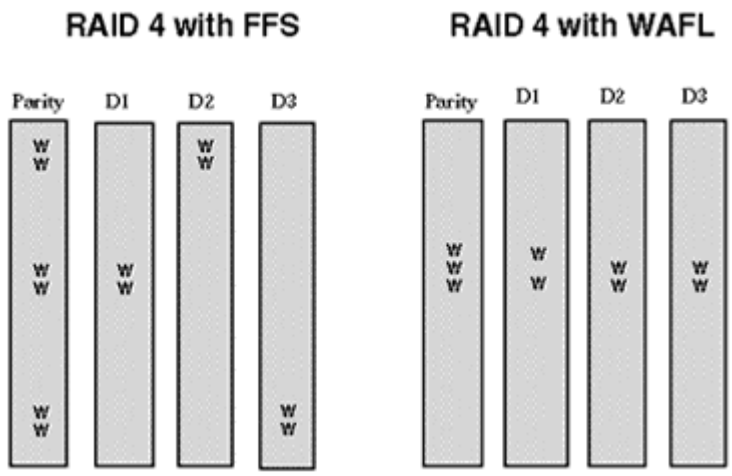


Figure 4: FFS and WAFL write allocation patterns.

Since the Berkeley FFS doesn't understand the underlying RAID4 layout, it tends to generate requests that are scattered over the data disks, causing the parity disk to seek excessively. WAFL writes blocks in a pattern designed to minimize seeks on the parity disk. Figure 4(b) shows how WAFL allocates these same blocks to make RAID4 operate efficiently. WAFL always writes blocks to stripes that are near each other, eliminating long seeks on the parity disk. WAFL also writes multiple blocks to the same stripe whenever possible, further reducing traffic on the parity disk. Notice that FFS uses six separate stripes in Figure 4(a), so six parity blocks must be updated. In Figure 4(b), WAFL uses only 3 stripes, so only 3 parity blocks are updated and they are all near each other.

As a WAFL file system becomes full it uses more stripes to write a given number of blocks which increases the number of parity blocks that need to be updated. Even in a very full file system, however, a small range of cylinders contains many free blocks, so the more important benefit of reducing seeks on the parity disk remains. Like FFS, WAFL reserves 10% of disk space to improve performance.

6. Conclusion

The goal of building an appliance file server led us to incorporate a collection of features not found together in file servers based on UNIX:

- Fast and simple system installation
- Fast system reboot, even after power loss or system failure
- Large partitions that grow as disks are added
- RAID to protect against disk failures
- Snapshot copies to simplify backup
- Simple administrative interface

These features, combined with the performance and reliability benefits of the appliance approach, result in a file server whose simplicity and price/performance cannot be matched by any general-purpose file server.

7. Bibliography

Sailesh Chutani, et al. The Episode File System. Proceedings of the Winter 1992 USENIX Conference, pp. 43-60, San Francisco, CA, January 1992.

Bob Lyon and Russel Sandberg. Breaking Through the NFS Performance Barrier. SunTech Journal 2(4): 21-27, Autumn 1989.

Marshall K. McKusick. A Fast File System for UNIX. ACM Transactions on Computer Systems 2(3): 181-97,August 1984.

John Ousterhout and Fred Douglis. Beating the I/O Bottleneck: A Case for Log-Structured File Systems. ACM SIGOPS, January 23, 1989.

D. Patterson, G. Gibson, and R. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). ACM SIGMOD 88, Chicago, June 1988, pp. 109-116.

Russel Sandberg, David Goldberg, Steve Kleiman, Dan Walsh, and Bob Lyon. Design and Implementation of the Sun Network File System. Proceedings of the Summer 1985 USENIX Conference, pp. 119-30, Portland, OR, June 1985.