ServerGuard: The Pinnacle of Continuous Data Access

Garth Gehlbach,
Joseph Giammarco,
and Auspex Engineering

Technical Report 12
Fourth Edition
October 1997

AUSPEX

Executive Summary

This paper describes ServerGuard™, a unique fault-tolerant and disaster-tolerant network filesystem (NFS) data service solution developed by Auspex, the leader in continuous data access. ServerGuard addresses the stringent high-availability needs of customers whose business is tightly linked to its information systems. These customers are typically searching for fault-tolerant solutions that can withstand component or system failures without disrupting normal operations. ServerGuard not only protects data from being lost or corrupted due to a failure; it also allows for continued data access in the face of a site disaster.

ServerGuard uses the network to send writes simultaneously to two servers, efficiently replicating data to keep two separate systems in full synchronization. This approach yields unmatched data protection and data availability, while boasting superior performance and minimal incremental network traffic.

ServerGuard benefits can be summarized as follows:

- **Data integrity is preserved** since data is synchronized and live; the client is acknowledged after both servers have data in stable storage
- **Latency is low and server performance is improved** because writes go to both servers simultaneously; alternative schemes require writes to be duplicated by the primary server
- **Network performance is high** because ServerGuard does not create a lot of extra network traffic
- **The highest level of protection** is provided by the "shared nothing" ("redundant everything") architecture with geographic separation between servers
- **Cost of ownership is reduced and installation simplified** in that ServerGuard uses existing, standard network connections
- **The value of the secondary system is extended** since the secondary system can provide file service in addition to participating in a ServerGuard mirror

ServerGuard is the premier solution for business-critical NFS data service, and the capstone of Auspex's portfolio of solutions delivering continuous data access.

Document 300-TC045, V4.0, 971020

Auspex Systems Inc.
5200 Great America Parkway
Santa Clara, California 95054 USA
Phone: 408/986-2000 Fax: 408/986-2020
Email: Info@Auspex.com
Web: http://www.auspex.com

1. Introduction

1.1 The Need for High Availability

High-availability products are becoming increasingly important in the computer industry. One of the fundamental reasons for this movement is the trend toward greater dependence of enterprises on serving and processing data every moment of every day. This is understandable since data, the lifeblood of a corporation, is an asset that must be accessible at every instant, around the clock, for the company to compete or even survive. Examples are abundant in the financial, telecommunications, and Internet industries, where losing data access means lost business and the loss of data can be catastrophic. Leading-edge companies spend considerable resources protecting and ensuring the availability of their precious data. These businesses also understand the impact of the unavailability of data. The more sophisticated firms may even measure data availability, or at least make information technology (IT) plans and investments based on estimates of availability.

1.2 Cost of Downtime

A popular yardstick for measuring the impact of availability is the cost of downtime. Promoted by analysts and vendors alike, this analysis tool is useful for planning and designing IT architectures and for selecting appropriate solutions in response to business needs. The cost of downtime generally factors in such elements as hard and soft costs, idle employees, loss of production, loss of opportunity, and customer dissatisfaction. For a more complete picture of the impact of downtime, it is also important to consider the unproductive time after an outage when the organization does not realize that service has been restored. In addition, it is important to take into consideration the time lost while employees scramble to restore lost work and regain a productive attitude. Bottom line figures vary from industry to industry and company to company, but many range from $10,000 to millions of dollars per hour.

With so much riding on the availability of data, improving availability and reducing downtime, even by a few hours a year, can often justify investments in IT solutions that satisfy requirements for high availability. Consider also the need to protect against site disasters (fire, earthquake, flood, explosion, sabotage, etc.), and the list of desired product attributes grows.

2. File Servers in a Client/Server Computing Model

Today’s most popular IT architectures follow the network-based, distributed, client-server computing model in which information assets are coordinated globally and distributed throughout the enterprise. These client-server architectures are optimized for the needs of their most intensive users, but are also fully integrated with other information systems and are accessible to all users and locations.

One of the key components in the modern IT client-server network infrastructure is the consolidated file-server or data-server, which stores large amounts of critical data and serves it to data-intensive, mission-critical applications via high speed networks. These applications are typically supported on hundreds of client workstations and application servers. High availability is imperative for consolidated file servers; and the following discussion of high-availability nomenclature and technologies is directed specifically to them.

3. High Availability Nomenclature

The term “high availability” has become overused and has evolved to mean different things to different people. In this report, “high availability” is used to define the entire category of system solutions intended to increase system availability, including everything from the most simple redundancy schemes to highly sophisticated remote mirrored servers. In an effort to enable a more precise discussion and analysis, discrete points along this spectrum are described in some detail.

3.1 Levels of High Availability

Table 1 on the next page defines a number of high availability terms as they relate to typical protection features and their impact on users[1]. In general, high availability schemes classified as “recovery” tend to involve a complete interruption of work and a distinct, often lengthy recovery process before operations are restored. “Resilient” is a term used to connote a very brief interruption after a failure when no distinct restoration process is required. “Tolerant” is defined as the ability to sustain failures without disrupting operations. “Disaster” is a term that implies a degree of physical separation of systems such that a site disaster will not impact the peer system.

[1] The definitions for levels of high availability are loosely based on IDC’s model describing four availability levels in the report “Highly Available Systems,” authored by R. Desautels, G. Lee, and S. Josselyn in September, 1996.

Availability Label	User Impact upon Component Failure	System Protection Features (Typical)
Disaster Recovery	Work stops; uncontrolled shutdown; data integrity ensured; data accessible after restore process; requires re-logon	Remote vault with backup copy of data
Basic Data Protection or Fault Recovery	Work stops; uncontrolled shutdown; data integrity ensured	Disk mirroring or RAID and a log-based or journal filesystem for identification and recovery of incomplete in-flight transactions
Fault Resilient	User interrupted; can quickly re-logon; may have to re-run some transactions from journal file; may experience performance degrade	User work transferred to backup components; multiple system access to disks
Fault Tolerant	User stays online; current transaction may need restarting; may experience performance degrade	Automatic failover transfers user session and workload to backup components; multiple system connections to disk
Disaster Tolerant	User stays online; current transaction may need restarting; may experience performance degrade	100% component and functional redundancy; automatic failover; geographic separation
Transparent and Synchronous Fault Tolerance	Transparent; no interruption of work; no transactions lost; no/little degradation in performance	100% component and functional redundancy; automatic failover; synchronous operation
Transparent and Synchronous Disaster Tolerance	Transparent; no interruption of work; no transactions lost; no/little degradation in performance	100% component and functional redundancy; automatic failover; Synchronous operation; Geographic separation

Table 1: Definitions of High Availability Levels

3.2 Fault Resilient and Disaster Recovery

As indicated in the definitions of terms relating to high availability, fault resilient and disaster recovery alone cannot meet the needs of users who require instantaneous access to their data at all times. This class of user cannot tolerate an interruption and lengthy recovery of data before access is restored. While fault resilience and disaster recovery solutions may be useful in protecting data, more stringent high-availability requirements focused on data access are better satisfied by fault-tolerant or disaster-tolerant solutions.

3.3 Fault Tolerant Characteristics

Fault-tolerant solutions generally strive to eliminate user disruption in the event of a failure. One approach to measuring the effectiveness of fault-tolerant solutions is to consider the "6 Rs" of high availability, namely:

- **Redundancy**—Single points of failure are eliminated
- **Recognition—**Faults are automatically detected and, since there are a great variety of possible faults, numerous mechanisms are usually required
- **Restoration of Service—**Once faults are determined, failovers should be quick to continue operations, and client action should not be required
- **Repair—**Isolation of failed components is necessary to enable non-disruptive repair

- **Resynchronization—**When a failed component is reinstated, data must be updated on the failed system or subsystem with the modifications that occurred since the failure
- **Return to Service—**Upon successful resynchronization, normal (protected) operations should be restarted, preferably automatically

Within the fault tolerant label, there may be finer distinctions that can be used to describe the relative impact to users and the risk of data loss. The most desirable attributes are transparency to users and complete synchronicity (data is written safely to stable storage on both servers or to more than one server before acknowledgment is given to the client).

4. Fault-Tolerant Approaches

The following subsections describe various approaches to fault tolerance at the system level. The solutions presented here are able to withstand virtually any component failure, and some may even address the more unlikely, but more serious, system failures and site disasters. Fault tolerance at the subsystem level is also worthy of note, particularly when considering hardware or software components that may exhibit relatively high failure rates. A list of such vulnerable elements of a system include application software, operating system software, disks, and power supplies. Solutions to protect against disk and power failures can be found throughout the industry, including Auspex's mirroring and RAID 5 DriveGuard offerings. However, for isolation to application and operating system failures, Auspex's DataGuard software is a standout solution.

4.1 Shared Disk

One of the most common implementations of fault tolerance is the "shared disk" cluster (see Figure 1 on the next page). This implementation involves two or more servers connected to a single disk subsystem. A key benefit to this approach is the relatively low premium associated with the increase in availability. This cost benefit is due to the use of RAID (Redundant Array of Independent Disks) configurations, which are often employed in place of mirroring, that require fewer disks for data protection than mirrored configurations. This level of protection, however, is not as good as the "shared nothing" architectures described below. Specifically, shared disk schemes using RAID have a single, albeit protected, copy of data, and may be vulnerable to corrupted data or catastrophic failures that affect the storage subsystem. Certain electrical problems on one system, for example, may prevent the other system from accessing the disks. This approach also restricts the separation distance between servers because of cable length requirements.

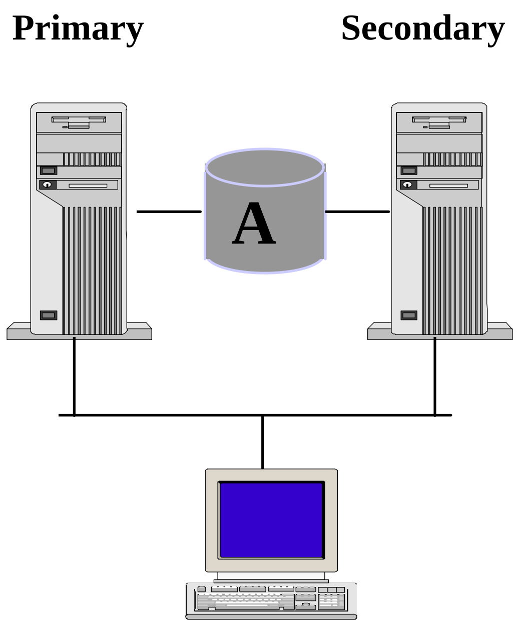

Figure 1: Shared Disk Configuration

4.2 Shared Nothing—Server Initiated Copy to Secondary

Another popular design in fault tolerance is the “shared nothing” approach in which data is copied to a peer system. This architecture involves a basic mirroring facility in the primary server where the mirrored disk is remotely located in the secondary server, as illustrated in Figure 2 on the next page. While offering good protection by virtue of its complete redundancy and potential tolerance to disasters depending on separation distance, there are a few exposures that may be inherent to the design. Since writes require the primary server to (1) process the write request, (2) generate a mirror copy, (3) send the copy to the remote secondary via a network connection, (4) wait for the secondary to complete the write and acknowledge back, (5) receive and process the acknowledgment, and finally (6) acknowledge the client, there may be a significant performance impact versus a server that is not running in this fault tolerant mode. In addition, shared nothing implementations, in which the primary initiates a copy to the secondary, may be vulnerable to propagating corrupt data.

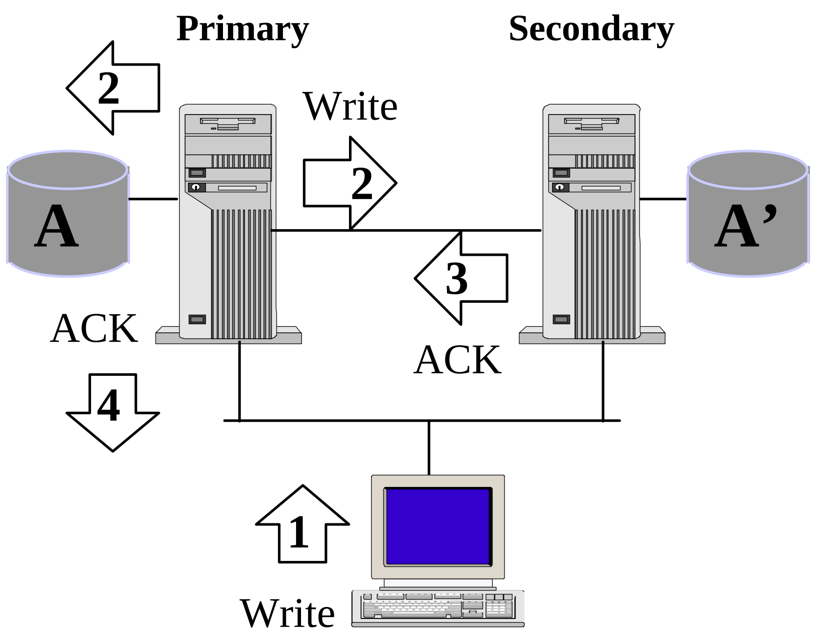

Figure 2: Shared Nothing Configuration—Server Initiated Copy to Secondary

4.3 Shared Nothing—Network-based Data Replication

A new approach to the shared nothing or mirrored server architecture is ServerGuard's network-based data replication. In this design, writes sent once on the network are received simultaneously by both primary and secondary servers for specified filesystems (see Figure 3 on the next page). The secondary server sends an acknowledgment to the primary server across a dedicated or non-dedicated "heartbeat" network link, and the primary server acknowledges the client. Compared to the previous example of a shared nothing architecture where the server initiates a copy to the secondary, network-based replication involves far fewer tasks, and tasks are performed in parallel. This translates into low latency (writes go to both servers simultaneously), minimal extra network traffic (data is only on the network once), and efficient use of the secondary server, which is able to provide other file services while acting as the failover system.

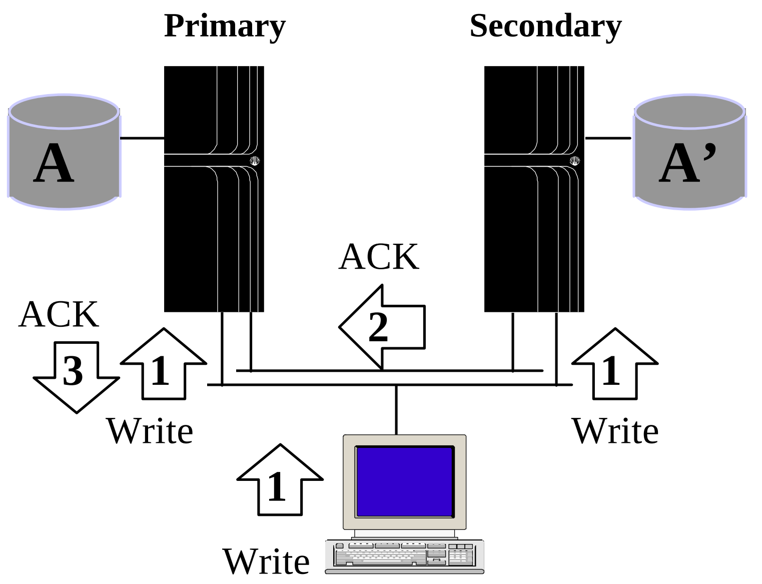

Figure 3: Shared Nothing Configuration—Network-based Data Replication

5. ServerGuard Overview

5.1 Network Filesystem (NFS) Attributes

NFS is a popular distributed filesystem protocol that has a client/server orientation. NFS is designed with the provision that, in the event of a prolonged loss of access to the file server, the client will continually retry its unsuccessful operations until service is restored. Once service is restored, the client will continue to operate without user intervention. While this adds to the convenience of writing applications that use NFS, many usage scenarios require higher availability than can be provided with one server.

5.2 The ServerGuard Premise

An NFS client normally issues a mount request to an NFS file server by specifying the server's hostname and the name of the filesystem the client wishes to access. In the ServerGuard environment, a virtual server hostname and IP (Internet protocol) address are defined to be used by clients that wish to access filesystems that are under ServerGuard protection. NFS calls made by clients to this virtual server are processed by both members of the ServerGuard pair. ServerGuard keeps both mirrored copies of the filesystem in synchronization, and is able to present to the client the illusion that the client is talking to a single server.

5.3 ServerGuard Technology Highlights

The virtual server is given a unicast IP address and a MAC (media access control) multicast address that is recognized by both members of the ServerGuard server pair. The client makes requests that are processed simultaneously by both servers, which through the use of a lightweight acknowledgment mechanism, are able to guarantee data redundancy and consistency.

If a network or system disruption causes loss of service to one of the two servers, the other will continue to process NFS requests alone until the disabled server or filesystem is brought back online. ServerGuard can then compare and copy data as needed in order to resynchronize the two copies of the protected filesystem.

The ServerGuard approach achieves the highest levels of availability: transparent and synchronous fault and disaster tolerance. ServerGuard's unique design differs from traditional two-phase commit algorithms (described above as the shared-nothing, server initiated copy scheme) in that it has lower overhead. It also differs from shared-disk solutions by allowing the servers to be physically separated. ServerGuard is a software-only product for Auspex NetServers. No special hardware, and no special client-based software are required.

5.4 ServerGuard Configuration

ServerGuard provides duplication of NFS filesystems across two servers connected by only a network. Each filesystem has one primary and one secondary server. The designations "primary" and "secondary" determine how each server handles NFS requests. For example, by default, only the primary will service a read request. (The newly introduced "local read" feature allows the secondary to service read requests for selected clients or groups of clients.) Since servers can have many filesystems, a given server may simultaneously act as the primary for some filesystems, as the secondary for others, and as a conventional standalone server for still others. Figure 4 below illustrates such a configuration, where Server 1 is the primary for filesystem A and secondary for filesystem B (noted as B'), and Server 2 is the primary for filesystem B and the secondary for filesystem A (noted as A'). Server 1 also has filesystem C which is not mirrored (ServerGuard protected) to Server 2.

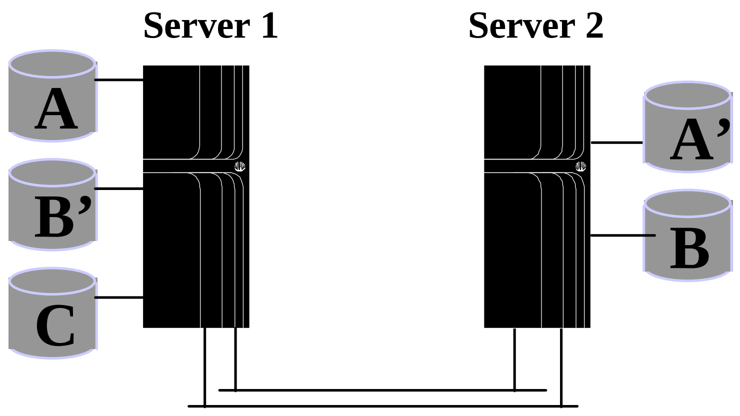

Figure 4: Sample ServerGuard Configuration

5.5 ServerGuard Benefits According to the "6 Rs"

ServerGuard's unique design enables the network to deliver data to a pair of mirrored servers, allowing ServerGuard to achieve high marks when measured on the high-availability criteria introduced in Section 2.3. Specifically, ServerGuard excels at the "6 Rs":

- **Redundancy—**ServerGuard has 2N redundancy, mirroring all components and eliminating any single point of failure
- **Recognition—**ServerGuard automatically detects faults, and provides a number of mechanisms to enable intelligent decisions based on the condition of the systems and network
- **Restoration of Service—**Failovers are quick and support continued operations; no client action is required
- **Repair—**By supporting simplex operation, ServerGuard can isolate failed components and provide for non-disruptive repair, all while providing clients continued access to data
- **Resynchronization—**ServerGuard can employ multiple algorithms to best address the task of swift resynchronization; ServerGuard also manages the resync process in multiple passes to minimize the read-only window while completing resynchronization quickly
- **Return to Service—**ServerGuard allows for a restoration of the original configuration of mirrored servers after completion of the resynchronization

6. ServerGuard Fundamentals

6.1 Multicasting and Virtual Servers

A pair of Auspex NetServers connected to a common network will each have, associated with their respective network interfaces, an IP address and a MAC address. When ServerGuard is installed on both systems, the two servers are able to create the image of a virtual server on the network. This virtual server has a normal, unicast IP address, so that NFS clients see it as a normal server. The MAC address associated with this IP address, however, is a multicast MAC address, enabling both NetServers to receive and process any packets designated for the virtual server. Because each server has its own copy of the ServerGuarded filesystem, and because all client requests for this filesystem are seen by both servers, the servers are able to keep the two copies of the filesystem in synchronization.

The multicast MAC address is algorithmically created from the unicast IP address that the system administrator provides. [Deering89].

The virtual server also has an NIS database and ARP (address resolution protocol) table entry.

6.2 ServerGuard Networking

ServerGuard currently includes support for Ethernet, 100BaseT, FDDI, and ATM (using LAN emulation).

The virtual server's filesystems can be exported over any or all of the subnets commonly connected to the server pair. Any subnet on which the virtual server is exported must be locally-attached to both servers. That is, multicasts on the exported subnets must, of course, be propagated to both servers. Bridges or virtual LANs (VLANs) may be used on a subnet containing a virtual server (see Figure 5 on the next page) providing they pass multicast MAC packets. It is also necessary for these connections to provide sufficiently low latency in order to avoid disruption to standard NFS time-outs and/or the ServerGuard fault detection protocol.

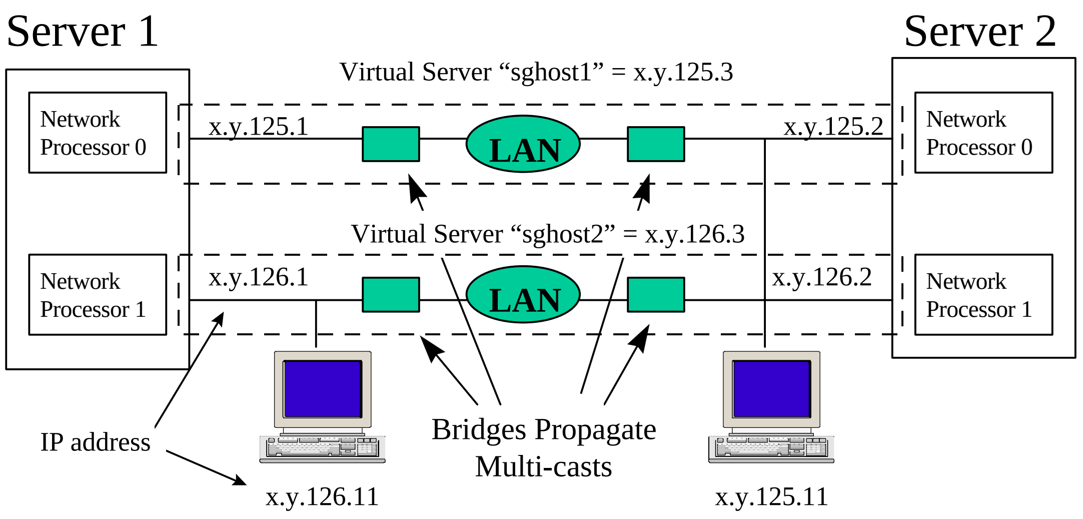

Figure 5: ServerGuard Over a Bridged Network

6.3 Synchronized Operation

At the core of ServerGuard is the mechanism for keeping the two sides of a mirrored pair in sync., which enables online, low-overhead, transaction-consistent (as defined by NFS) data redundancy between distributed servers. ServerGuard does this by "understanding" the implications of specific NFS request types with respect to data modification and client statelessness. The primary and secondary servers use this information to determine their actions for each class of NFS operation.

NFS operations can be classed in three categories [NFS89]:

- **Read operations**: read, getattr, null, lookup, readdir, readlink, statfs

  These routines do not modify the filesystem data contents, though some update the access time in the filesystem's metadata.
- **Write operations**: write, setattr, rename, rmdir, rm, link

  These routines modify existing files and metadata.
- **Create operations**: create, mkdir, symlink

  These routines result in new inode allocations.

6.3.1 Read Operations

By default, read operations are performed by only the primary server. ("Local reads" is an exception in which the secondary server can be configured to respond to read requests coming from certain selected clients.) As Table 2 below shows, the primary replies to the client with the requested data. The secondary server receives read requests, but only updates the affected inode's access time. Since the file times are independently updated, the public domain Network time protocol (NTP) is run to synchronize the two servers' clocks. Nevertheless, atimes/mtimes/ctimes on the two servers may be slightly different from each other, but the relative order of atime/mtime/ctime between files is guaranteed to be the same for non-overlapping requests. It is the relative order of these time stamps that is critical (for example, *make* and similar applications) rather than the actual time stamp values.

Primary	Secondary
1. Receive request from client; create DRC entry 2. Perform request 3. Reply to client	1. Receive request from client; create DRC entry 2. Update file's access time

Table 2: ServerGuard Processing of Read Requests

Because read operations constitute the majority of a typical NFS request mix (81%), ServerGuard secondary servers do much less work than primary servers. This is one of the reasons that ServerGuard designates primary/secondary roles on a per-filesystem basis. For a pair of servers, each could act as primary for half the filesystems, thereby providing static load balancing. With the use of the "local read" feature, load balancing can also be achieved for a single filesystem.

6.3.2 Write Operations

Table 3 describes ServerGuard's treatment of write operations.

Primary	Secondary
1. Receive request from client, create DRC entry 2. Perform request 3. Sleep on DRCDRC entry waiting for ACK 4. ACK receive interrupt awakens the NFSD sleeping on DRC entry 5. Reply to client	1. Receive request from client, create DRC entry 2. Perform request 3. Send ACK to Primary

Table 3: ServerGuard Processing of Write Requests

The write operation indicated assures that when a reply is sent to the client, the write has been performed on both servers. If something fails and a reply is not received by the client, the NFS client will time out and retry the operation. A more detailed discussion of failure modes and responses is covered in Section 6.4 and Section 7.

6.3.3 Create Operations

Operations that allocate inodes are treated separately from writes because NFS clients retain state information in the form of file handles. File handles are opaque data types that in many NFS implementations contain:

- Filesystem ID
- Inode number
- Generation number

To shield the client from knowing that a server failover has occurred, all file handle contents must remain identical across both servers. Filesystem ID consistency is achieved by statically defining logical filesystem IDs (rather than using a physical tag, such as the major/minor device number) when making a new filesystem. Inode/generation numbers, which are randomly generated and are therefore different on the two servers, are instead kept identical by the create protocol.

To handle creates, ServerGuard first performs the operation on the primary server and sends the resulting inode/generation pair to the secondary server, which then “allocates” that specific inode (see Table 4 for a summary of ServerGuard processing of creates). Since this is done serially, creates have a response time that is twice as long as usual (although not much more work is actually done by the system). In typical workloads, creates amount to only 2% of the request mix, which means the overall performance of the ServerGuard pair is minimally impacted by the longer response times of creates.

Primary	Secondary
1. Receive request from client, create "duplicate request cache" DRC entry	1. Receive request from client, create DRC entry
2. Perform request	2. Sleep on DRC entry
3. Send inode/generation to secondary server	(Sleeping)
4. Sleep on DRC entry	4. ACK interrupt inserts inode/generation in DRC and wakes up the NFSD
(Sleeping)	5. Create the file using primary server's information
(Sleeping)	6. ACK the primary
7. ACK receive interrupt awakens the NFSD sleeping on DRC entry	
8. Reply to client	

Table 4: ServerGuard Processing of Create Requests

6.3.4 Duplicate Request Cache (DRC)

Standalone NFS server implementations possess a data structure called the "duplicate request cache" (DRC). The DRC contains an entry for each in-progress request as well as for the most recently completed requests. The DRC's original purpose was to provide correct behavior for non-idempotent NFS procedures such as create and remove. ServerGuard uses the DRC as a placeholder for state information pertaining to individual NFS requests.

6.3.5 Time Stamps

Although the data in both copies of a ServerGuarded filesystem will be identical, time stamps of files in the filesystems can be different. There are several reasons for this:

1. The clocks may be slightly out of synchronization. ServerGuard includes the public-domain Network Time Protocol (NTP) to keep clock skew within 3 seconds. Other clock synchronization utilities may be used instead.
2. One server may take longer to complete the request than the other server.
3. A request may be performed on a retry by one server. A network fault may have caused a request packet to be received by only one server. That server would have performed the request on the initial transmission, while the peer would do it on the retransmission at a later time.

Note that, when a client looks at files and time stamps in a filesystem, the client is reading these from the filesystem's copy on the primary server. In the event of a failover to the secondary filesystem, the time stamps will now come from the secondary server's copy of the filesystem and will therefore be slightly different.

The relative order of time stamps on both servers is guaranteed to be identical for any pair of serialized NFS operations. Any two operations are considered serialized if their execution time does not overlap at all. In other words, the reply to one is received before the request for the other is sent.

6.3.6 Access Times

Since ServerGuard processes reads without any handshaking between the servers, it is possible for the secondary to miss access time updates. The exposure is relatively insignificant in that:

- Files are usually read using multiple closely-spaced NFS operations; the chances of the secondary missing all of them are very rare
- A filesystem checking utility is provided to peruse the inodes overnight and propagate any access time updates that were missed
- The files on the secondary are not seen until a failover is needed

6.3.7 Directory Ordering

The contents of directories on both copies of the filesystem are guaranteed to be the same. However, the *order* of the directory entries *may* be different.

Again the exposure is minor, the only consequence being that if an NFS client is in the middle of a *readdir()* sequence when a failover takes place, it may receive an error as a result. A retry of the *readdir()* from the beginning of the directory would resolve the problem (if this retry is accommodated in the client application code).

Directory ordering can be different only if files in a directory are created in non-serialized fashion. That is, multiple clients attempt to create files at the same time.

6.4 Transient Network Faults

Because of the connectionless nature of NFS (UDP) and ServerGuard, ServerGuard is able to recognize and handle transient network faults transparent to the client.

Consider the case where a client NFS write request reaches the primary, but is dropped by the network before reaching the secondary.

The primary performs the write operation, then sleeps on the DRC entry. After 1 second, the NFS client will reissue the request, since there was no response the first time around. The primary will ignore the second request (as a duplicate) and continue to sleep awaiting an acknowledgment ("ACK") from the secondary. If the fault was temporary, the secondary will receive the second request and perform the operation, then ACK the primary. The primary will awaken upon receipt of the ACK and send a reply to the client, completing the request. See Table 5 below for a summary of steps ServerGuard follows for transient network faults. Responses to problems which are not transient are described in Section 6, Fault Recognition.

Primary	Secondary
1. Receive request from Client, create DRC entry	1. No request received
2. Perform request	
3. Sleep on DRC entry	
4. Receive request (retry) from Client; ignore	5. Receive request (retry) from Client
5. (sleeping)	6. Perform request
6. (sleeping)	7. ACK the primary
7. ACK receive interrupt awakens the NFSD sleeping on DRC entry	
8. Reply to client	

Table 5: ServerGuard Handling of Transient Network Faults

7. Fault Recognition

ServerGuard automatically recognizes events that necessitate failovers. Such events can be component failures, complete server failures, software panics, network faults and so on. Any time a server detects an impaired client access, some form of action is taken.

Depending on the nature of the fault, there are two forms of failover: declarative and symptomatic.

7.1 Declarative

A declarative event is one in which the failing component is able to notify the peer server of its distress and intent to cease operation. Examples of declarative events include software panic, disk failure, individual board failure, or operator shutdown.

Failover is swift and immediate since no "think time" is involved. In fact, tests show failover times of less than one second for declarative failovers.

7.2 Symptomatic

Symptomatic events are those in which the failing component is not able to notify the peer of the anomaly. In fact, the cause of the problem may be external to the server altogether (for example, network cable breakage or sudden power loss).

While failovers can take as little as a second to effect, the decision to fail over (on symptomatic events) can take 30 to 60 seconds (depending on user-tunable parameters). By configuring ServerGuard to take longer to effect a symptomatic failover, premature failovers can be avoided. For example, a transient pause in communication service should not cause a failover if the user-tunable parameters are properly set.

The three "symptoms" consulted are:

- **Heartbeats—**ServerGuard servers pass heartbeat messages between each other, at a default frequency of one a second; if a heartbeat is not received after 10 seconds, an alert is raised
- **ServerGuard peer ACKs—**When a primary ServerGuard server performs a write or create request but does not receive an acknowledgment (ACK) from the secondary within 5 seconds, an ACK time out event is recorded
- **Client NFS retries—**The duplicate request cache mechanism allows servers to detect when an NFS request from a client is retried; if an unusual number of retries are detected, an alert is raised

Because the determination of symptomatic failovers includes an element of uncertainty, a software "hook" mechanism, ax_fthook, is provided to allow users to override the failover decision. ServerGuard can be made to call a user-supplied program, pass it the parameters of the three symptoms used in the decision making, then expect a yes/no answer in return. The user-supplied program is flexible in the types of things it might check and analyze before triggering a failover. Among other things, this may include any one of the following:

- Operator confirmation (solicited by a system initiated page)
- Checking other system indicators
- Checking the time of day

8. Failover

When the decision has been made to invoke a failover for a filesystem (or for all filesystems, in the case where an entire server becomes disabled), the server with the "failed" filesystem stops responding to NFS requests for that filesystem. Client NFS requests continue to be multicast to both servers (as the client is unaware of the failover) but there are no ServerGuard peer ACKs between the servers for that failed-over ("simplex") filesystem.

Once a failover is effected, the disabled component (filesystem or server) is available for servicing.

9. Resynchronization

When the "failed" server or filesystem is repaired and made available, the "failed" server contacts the surviving server. The failed server then verifies that the survivor has been operating in standalone ("simplex") fashion and initiates a utility to update the failed server's disk contents. ServerGuard refers to this process as resynchronization. Resynchronizations are performed on a filesystem-by-filesystem basis.

9.1 Time Stamp Copy

Resynchronization involves the logical process of creating, on the failed server, an identical copy of the data found on the surviving server.

Resynchronization may involve copying all the files that were written/updated while the filesystem was in simplex (standalone) mode. These files are identified by scanning their inode modification/change times (mtime/ctime).

For those files that were merely read, but not changed or updated, during the simplex period, only their inode contents are copied.

Information about free inodes is also passed to the target (failed) server, so the target server can free any such inodes that are still allocated. This reflects user file deletions that occurred during the simplex period.

In some cases, it is more efficient to copy the entire surviving system’s filesystem to the failed system’s copy of the filesystem, using dd. This “dd resynch” may be invoked by the resynchronization process when, for example, the source filesystem is fairly full and ServerGuard determines that the dd will be faster than the normal compare/copy process. The dd resync may also be invoked if the failed filesystem is empty and the surviving filesystem is not.

9.2 Fast Resync

ServerGuard has recently been enhanced to greatly increase the speed of large file resynchronization. For file sizes larger than 45 KB, the data copy mechanism is file transfer protocol (FTP) via the network processor (NP), which yields data transfer rates three times the alternate scheme employing the host processor (HP). Due to the overhead of FTP in setting up file transfers, however, a performance gain is not realized for smaller file sizes. ServerGuard automatically chooses the best method of data transfer, based on file size, and uses that method for resynchronization.

9.3 Multiple Passes

While a resynchronization is taking place, clients continue to try to access and change the source filesystem. Unless clients are prevented from changing the source filesystem as it is being scanned and copied during a resynch, it is impossible to guarantee that the source and target will be identical at the end of the compare/copy resynch pass. The most straightforward way to resynchronize the system is to “lock” the filesystem from client access while the resync is underway. However, this results in an unacceptably long period of filesystem unavailability. To decrease this window of unavailability, ServerGuard performs resyncs in multiple passes. During the second pass, the filesystem is put into read-only mode (“locked”). Write and create operations from clients initiated during the second pass will not complete until the end of the second pass.

For example, consider the case where a server has been down for almost 3 hours (10,000 seconds). During that particularly busy period, clients have updated files totaling 1 GB in size on the standalone server's copy of the filesystem (representing an update rate of 100 KB/second). In this example, let us assume an FDDI connection between the two servers.The first resync pass (unlocked) might take 12 minutes to complete. During that 12 minute period, another 72 MB of updates will be accomplished (again assuming an update rate of 100 KB/second). The second pass (locked), will then take approximately 1 minute to copy the 72 MB of new data. Therefore, the multiple pass approach would reduce the read-only lockout time by an order of magnitude.

9.4 Overnight Check

A variant of the resync utility is provided to verify and ensure the consistency of the two copies of a filesystem at any time. To be fully effective, this checker should be run with the filesystem locked from user access, allowing it to run either weekly or nightly at a suitably idle time for the customer.

9.5 Deferred Resync

ServerGuard provides administrators with the option of performing resyncs as soon as a failed server returns. Since the surviving server is providing complete NFS service, some sites may elect to defer the resync until a quieter time. This would reduce the impact of the network bandwidth usage and the read-only “locked” period incurred by resyncing.

9.6 Split Brain

“Split brain” refers to the case where the two ServerGuarded servers become separated and are unable to communicate with each other due to a network failure (see Figure 6 on the next page). If each server is still able to communicate with the clients on its side of the “break,” it is possible that the two copies of a ServerGuarded filesystem will be changed and updated independent of each other. This independent change and update occurs because each server thinks that the other is down and is therefore assuming the simplex role. When service is restored, both servers recognize that each of them had allowed their copy of the filesystem to be updated by their local clients. The filesystem now exists in two versions that, although different, are both good. Reconciliation of the data between these filesystems can be a difficult job.

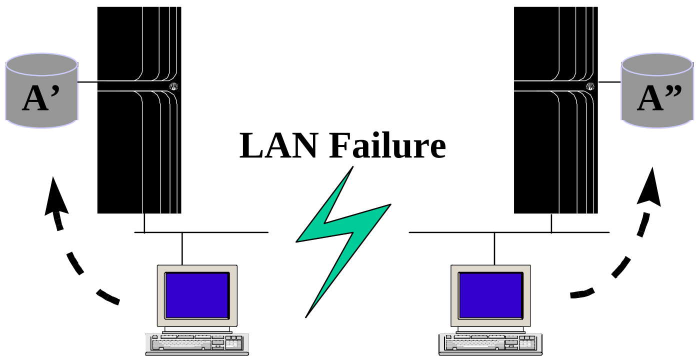

Figure 6: Split Brain Situation

ServerGuard is able to avoid split brain situations by offering a default option that allows the secondary server shift into read-only mode in the event of a symptomatic failover. (A declarative failover implies a communication path between the machines, so the normal, immediate failover will take place.) In this way, only one side can be updated. The administrator can set the secondary server to standalone read/write if the primary server has truly crashed. The disadvantage of this approach is that on symptomatic failovers, the filesystem is read-only until the operator intervenes.

Users can override this option and choose instead to have the secondary server shift to standalone read/write immediately upon symptomatic failover. In the rare event that a split brain occurs, when service is restored, ServerGuard will select one server to remain on line, and the other will be brought off line. The administrator must then manually reconcile the two versions of the filesystem before duplexed operation can begin. Alternatively, ServerGuard can be configured to resync the off-line copy from the one left on line. This results in the loss of updates on the ServerGuard server that was taken off-line.

9.7 Return to Service

At the completion of resync, the servers return to duplex operation, with each server resuming its original role. That is, if the secondary has been acting in standalone fashion, it will return to the secondary role.

10. Performance

ServerGuard is more efficient than other distributed solutions because:

- The original client request is multicasted; thus, a single network operation allows the data to be efficiently delivered to both servers
- The ServerGuard protocol is extremely lightweight; by relying on the intrinsic retry capabilities of NFS and by separating operations into read/write/create, ServerGuard reduces protocol overhead to 0, 1, and 2 messages respectively
- Resyncs are efficient, and filesystem lockout time is greatly reduced with the multiple pass approach

Table 6 quantifies performance for a typical workload mix (81% reads, 17% writes, 2% creates). The NFS block size used in the computation is 8 KB, rather than the 2 KB used in the official SpecNFS benchmark, to more closely reflect typical customer environments. Real configurations may deviate to some extent from the typical workload mix used in this model.

Performance Impact				
Aspect	**Total**	**Reads**	**Writes**	**Creates**
Latency	*+4%*	*No change*	*+10%*	*+100%*
Network Traffic	*+4%*	*No change*	*+1 packet (was 7)*	*+2 packets (was 2)*

Table 6: Performance Analysis Using a Typical Workload Mix

10.1 Latency

A ServerGuard pair has average response times that are 4% faster than a standalone server. The relationships between read, write, and create operations and latency are as follows:

- Read operations have no latency difference between ServerGuard and standalone because no ServerGuard handshaking is involved; the primary server simply processes the read request and replies to the client
- Write operations have a latency increase of approximately 10% because of the single ACK packet sent from the secondary server to the primary server
- Create operations are effectively doubled in latency because the create must be done on the primary server first; the create can then only be done on the secondary using the resultant inode/generation pair
- A typical workload mix of 81% reads, 17% writes, and 2% creates accounts for the composite 4% net impact to response time

10.2 Latency Across Geographically-Distributed Networks

When separating the server pair across a VLAN (geographically separated) or MAN, it is recommended that the "local read" feature be implemented. Read requests will then be satisfied by the local server. Write and create operations involve handshaking. Since writes and creates amount to 19% of the typical workload mix, 19% of response times will increase by the average round-trip latency. When clients experience a failure of their local server, all accesses will be satisfied by the remote server, increasing response times by the average round-trip latency.

10.3 Network Bandwidth

ServerGuard ACK packets and inode/generation messages account for a 4% increase in network usage (as measured in Ethernet packets). The effects of reads, writes, and creates on network usage are as follows:

- Reads have no handshake and thus no increase; standalone NFS uses one packet for the request, and six for the reply
- Writes have one ACK packet added by ServerGuard, in addition to the six standard packet request and one packet reply
- Creates have two additional packets attributed to ServerGuard (one for the inode/generation and one for the ACK); standalone NFS uses one packet for the request and one for the reply

10.4 Throughput

A pair of ServerGuard servers[2] will yield 40% throughput improvement over a single server. Put another way, two standalone servers would yield 2*x* the throughput of a single server, while a ServerGuard pair yields 1.4*x*. The throughput cost of ServerGuard redundancy is thus 30% [(2-1.4)/2].

Writes and creates are basically done by both servers, and take a little more work because of the handshaking involved.

Reads on the other hand, are done only by one server (the primary by default, or by the secondary for some clients if Local Reads are enabled.) The server that doesn’t service the reads will merely update the file's in-core inode access time. This reduced effort in Reads more than makes up for the write/create handshaking overhead.

Applications with higher Read percentages than the typical workload mix described in this report will experience even better throughput performance.

11. Lock Manager

A utility closely related to NFS is the Lock Manager. ServerGuard provides a high availability version of the Lock Manager using a different redundancy scheme from the NFS protocol.

In normal NFS, when a client requests a lock and the request is granted by a server, the result is stored in two places:

- The client’s lock table of locks held by the individual client
- The server’s global lock table of locks held by all clients

Both these lists are kept in memory (unstable storage) because each is frequently updated. The server keeps a separate list (in stable storage) of all the clients that hold a lock or locks on that

[2] The ServerGuard pair is assumed to be balanced, where each is primary for some filesystems and secondary for others.

filesystem. The client's list is much smaller and is updated only the first time a client acquires a lock on that filesystem.

When a standalone Lock Manager server reboots, it rebuilds the global lock list by contacting all the clients (held in the stable storage client list) to obtain their subset lock lists.

The ServerGuard version of Lock Manager takes advantage of this redundancy of having the lock lists held by the server and by the clients in aggregate. Normally, the primary server for a filesystem services all lock requests. The individual lock requests are not reflected on the secondary at all.

The only communication between the primary and secondary happens when the list of clients is updated. The primary notifies the secondary of the update, and does not grant the lock until the secondary responds. In this way, the list of clients (but not the global list of locks) is mirrored between the servers.

When a Lock Manager failover occurs, the secondary contacts each on the list and builds the global lock list from their local lock lists. This technique turns out to be quite efficient, since the client list is not updated as frequently as the global lock list.

12. ServerGuard: Network-based Data Replication for Continuous Data Access

ServerGuard continues to be unique in its approach to high availability. No other solution can match its benefits of transparent and synchronous fault tolerance and disaster tolerance. ServerGuard's network-based data replication means that writes are sent to a pair of servers simultaneously; the data traversing the LAN makes one trip to get to both servers. The result is an efficient, low latency solution that provides the highest levels of data protection and continuous data access.

13. Glossary

ATM	Asynchronous transfer mode. A cell-based (53 bytes) network technology that provides very high bandwidth (commonly 155Mbps) and low latency.
ATM LANE	ATM LAN Emulation. A suite of protocols and standards that allows current hosts to take advantage of ATM without significantly modifying operating systems or application software (i.e., makes ATM look like ethernet or token ring). ATM LANE currently supports token-ring and ethernet emulation.
DRC	Duplicate request cache. A data structure that has one element for each of the most recent NFS requests. Present in most NFS implementations, but extended to add ServerGuard fields.
Failover	The transition of a ServerGuard server from its duplex (redundant, mirrored) role into standalone operation.
File handle	An opaque data structure that NFS clients use to reference files after opening them. In the case of Auspex's NFS implementation, the file handle is a composite of the filesystem ID, the file's inode number, and the inode's generation time.
Generation	Whenever a file is deleted, its inode is eligible for reuse. When the inode is reallocated, the time stamp of the allocation is recorded in the inode. This protects mistaken identity accesses from clients that still might have file handles for the deleted file.
Inode	Index node. Most NFS filesystems internally identify files by their index within an array of header structures, one per file, at the beginning of the filesystem.
Idempotent	An operation which yields the same result even when repeated. For example, writing the value “5” to a variable multiple times leaves the same result. Performing a non idempotent operation such as addition yields different results each time if repeated.
IP	Internet protocol. The traditional networking protocol for NFS traffic. Addresses are four bytes long, for example, 140.49.254.3.
LAN	Local area network
MAC	Media access control. The addressing mechanism used at the data link level by various LAN media. In ethernet, the MAC address is a 48-bit entity. Unicast, multicast, and broadcast capabilities are encoded by using specific ranges within the 48-bit address.
MAN	Metropolitan area networks
NFS	Network file system. A widely used remote file access protocol. NFS is

	capable of interoperating between many different types of operating systems.
Resync	Resynchronization. The process by which a returning server is brought up to date with filesystem updates from the surviving server.
Split brain	A condition in which a network partition has caused both servers to believe that their peer is down and revert to operating in standalone read/write fashion.
VLAN	Virtual-LAN. Creating multiple LAN-like network topologies through management software running on a network switch.

14. References

[Auspex97]	Auspex Systems Inc. *An Overview of Functional Multiprocessing for Network Servers.* Technical Report 10, 3rd Revision, Auspex Systems Inc., April 1997.
[NFS89]	Sun Microsystems, Inc. RFC 1094—*NFS: Network Filesystem Protocol Specification.* Network Working Group, March 1989.
[Deering89]	Steve Deering RFC 1112—*Host Extensions For Multicast Mapping.* Network Working Group, August 1989.