Saya Systems Inc.
2963 E. 3175 S.
Salt Lake City, UT 84109
sayasystems.com

ATLAS

A High Availability Server Designed Especially for Small Business

Michael P. Zeleznik, Ph.D.

The ATLAS High Availability Server (HAS) provides a cost effective means of ensuring that critical business services will continue to function in the event of hardware failures or planned server maintenance. Other existing HAS products come with either high upfront costs, or high ongoing support costs. Such high costs tend to be prohibitive for most small businesses, leaving them at the mercy of server failures and downtime. What has been missing is an HAS that is cost effective for small business. ATLAS (patent pending) is designed specifically to fill this need, tailored to the realities of the small business environment.

The purpose of this paper is twofold.

1. Evaluate the existing HAS products in terms of their design, function, and ongoing cost and effort required to keep them working properly. This illustrates the fundamental reasons for their high costs, upfront or ongoing.

2. Present an overview of the ATLAS architecture, comparing it with existing HAS products and describing how ATLAS provides a cost effective solution that is not possible with the existing products.

TABLE OF CONTENTS

1 Introduction

Today, most companies are dependent on computer technology to perform critical business functions. Workflow processes generally rely on one or more servers to provide essential services such as document management, database or portal access, or email. When a server fails, workflow processes will fail, often in unpredictable ways. Makeshift processes must then be implemented, inevitably resulting in reduced productivity and lost profits. This can also lead to customer dissatisfaction or even loss of customers.

For these reasons, large businesses expend substantial resources to minimize server downtime by utilizing what are commonly referred to as High Availability Servers (HAS). These fall into two categories: high availability (HA) clusters and non-stop computers. HA clusters are much less expensive than non-stop computers to purchase, but require substantial ongoing IT resources to ensure that they will work correctly at "failover." That is, when recovering from a failure. Conversely, non-stop computers require minimal ongoing IT resources to ensure correct operation at failover, but are much more costly to purchase.

Such high costs, whether up front or ongoing, tend to be prohibitive for most small businesses, leaving them at the mercy of server failures. What has been missing is an HAS solution that is cost effective for small business.

The ATLAS High Availability Server is designed specifically to fill this need. ATLAS ensures that critical business services continue to function in the event of hardware failures or planned server maintenance, while minimizing both the upfront and ongoing costs. ATLAS is of value to any business that prefers to put resources towards other profitable endeavors, rather than towards the ongoing costs of an HA cluster or the high purchase price of a non-stop computer.

The tables on the following page provide a summary comparison of ATLAS with other existing HAS products. The systems and issues in those tables are examined in detail in the remainder of this paper.

ATLAS consists of two commodity servers coupled with the ATLAS Sentry™ software, which provides a safe and reliable manual-failover environment that ensures ATLAS will work correctly at failover. In the event of a failure, one simply swaps the front mount RAID disks between the two servers and reboots. ATLAS currently supports services that run under Linux, and a Windows version is under development.

ATLAS evolved from the need for an HAS in a commercial, clinical environment. Since a non-stop computer was too costly, an HA cluster was the only option. However, the site could not provide the level of meticulous IT support that is required to maintain it. As such, an HA cluster may well have caused more problems than it would have prevented, and would have provided a false sense of security. Since no other cost-effective alternative existed, the need for a new solution was clear.

2 Comparison of ATLAS and Existing Products in terms of Failure Modes, Cost, and Function

Comparison of Failure Modes for All Options

Result / Failure	Hot Swap Server	Non-stop Computer	HA Cluster	ATLAS
Power supply fails	UP	UP	UP	**UP**
Fan fails	UP	UP	UP	**UP**
Disk fails	UP	UP	UP	**UP**
CPU fails	DOWN	UP	UP	**UP**
Memory fails	DOWN	UP	UP	**UP**
Motherboard fails	DOWN	UP	UP	**UP**
I/O Bus fails	DOWN	UP	UP	**UP**
Disk controller fails	DOWN	UP	UP	**UP**
Net interface fails	DOWN[1]	UP	UP	**UP**
Other hardware fails	DOWN	UP	UP	**UP**
Planned maintenance	DOWN	DOWN	UP	**UP**

Note 1: Can be UP if it is possible to configure with multiple NICs and failover.

Comparison of Cost and Function of High Availability Options

Result / Cost, Function	Non-stop Computer	HA Cluster	ATLAS
Cost (system)	High	Low	**Low**
Cost (IT resources)	Low	High	**Low**
Type of failover	Auto	Auto	**Manual[3]**
Danger of split brain?	No	Yes[2]	**No**
Confidence at failover?	Yes	No[1]	**Yes**
Verification of services?	No[1]	No[1]	**Yes**
Synchronization?	Yes	No[1]	**Yes**
Hardware failures	UP	UP	**UP**
Planned maintenance	DOWN	UP	**UP**
Load sharing possible	No	Yes	**No[3]**

Note 1: Possible only if the end user provides for this.
Note 2: Potential loss of data.
Note 3: These increase confidence at failover.

ATLAS provides the high confidence level and low IT overhead of a non-stop computer, while also supporting planned maintenance, at an affordable price. ATLAS continuously synchronizes the primary and secondary environments, and verifies that services will run correctly at failover, thereby removing that substantial, ongoing IT support burden. In addition, the ATLAS “reduced complexity architecture” with manual-failover increases confidence at failover, while further reducing the IT support requirements.

3 An Overview of ATLAS

3.1 Increased Confidence and Reduced Complexity

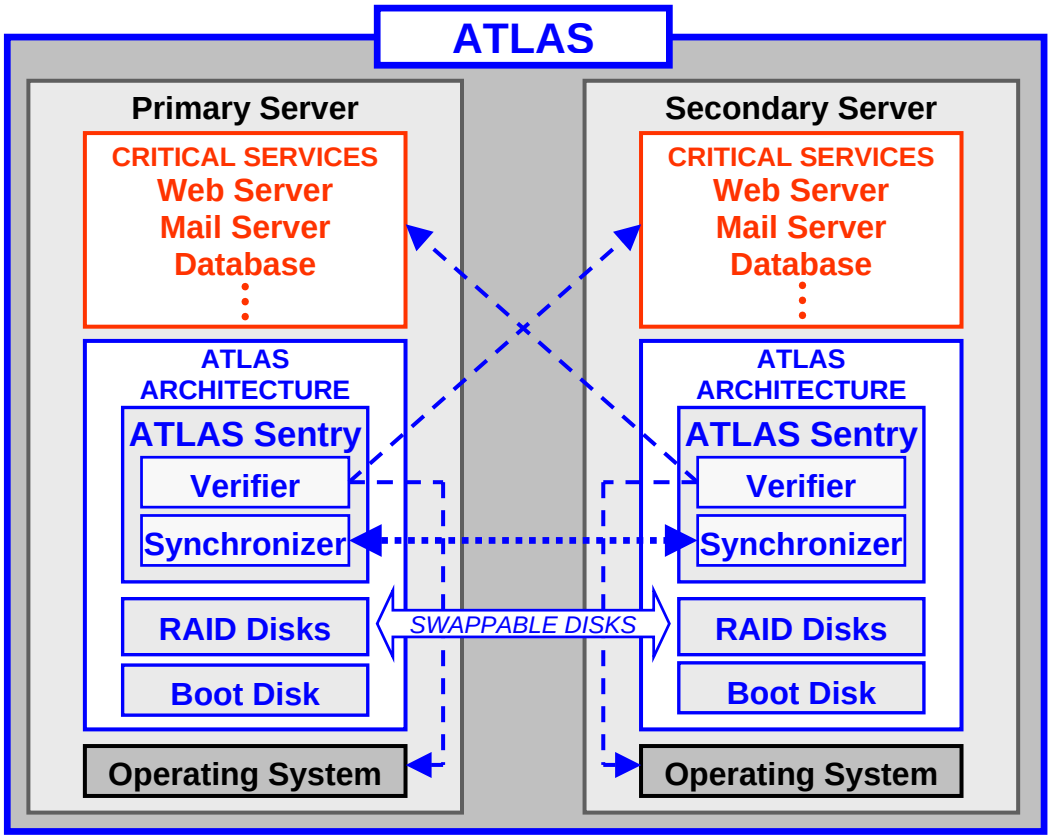
This discussion refers to the figure on the next page.

ATLAS is a cluster of two commodity servers (a primary and a secondary) with front mounted swappable RAID disks, connected to the existing business network. The primary runs all critical services (e.g., web server, email server, database server, file server) and stores all critical data.

If a problem occurs, one simply shuts down both servers, swaps the RAID disks from the primary to the secondary, then boots the secondary. The secondary now boots up as the primary, running all services and utilizing the same data and configurations as the original primary. We refer to this as “manual-failover.”

Fundamental to ATLAS is its “reduced complexity architecture” with manual-failover. Compared with existing HA clusters, this has fewer options and permutations to configure, maintain, and test over time, which reduces the required ongoing IT support. This also decreases the likelihood of problems at failover, since there are simply fewer things that can go wrong. Moreover, the cold restart ensures that all services start in the same manner and with the same system state as they did originally.

Also fundamental to ATLAS is its Sentry software suite. This ensures correct operation at failover by performing two critical tasks: (1) Sentry Synchronizer maintains synchronization between the primary and secondary, and (2) Sentry Verifier continuously verifies that all services run correctly on the secondary, as well as on the primary. These essential mechanisms relieve the end user of this substantial, relentless burden that exists with existing HA clusters. That is, ensuring that it will work correctly at failover. This significantly reduces the amount of ongoing IT support required, while increasing confidence at failover.



This figure is discussed on the previous page.

3.2 Preventing Downtime from Failures

ATLAS ensures that a hardware failure on the primary server will not disrupt services for more than the few minutes required to switch to the secondary. Examples of hardware failures include the CPU, motherboard, memory, I/O buses, internal cabling, controllers, disk drives, power supply, network cards or patch cables. A tri-server system can be used to insure against simultaneous hardware failures on both the primary and secondary.

If a problem occurs, one should simply switch to the secondary to see if it goes away. One need not have any idea of what the cause may be. If the problem disappears, then it was likely something in the primary and can now be investigated while business operations continue, unaffected. Resolving such problems, especially intermittent ones, can take hours if not days. During this time, business is not interrupted. Moreover, the problem can be investigated and resolved while not under intense pressure to restore services, which often results in makeshift solutions or “hacks” that can result in future problems.

3.3 Reducing Downtime for Maintenance

ATLAS can reduce downtime for planned maintenance, such as OS upgrades or new software additions. The maintenance can be performed on the secondary server while the primary server continues to function normally. If problems are encountered (as is often the case), they can be resolved while business functions continue unaffected. Once the secondary is working correctly, the roles can be switched. The secondary becomes primary and all clients now see the new system. Maintenance can then be performed on the other server.

4 Existing High Availability Servers

The significance of ATLAS, specifically in terms of its reduced IT support requirements and increased confidence at failover, can only be appreciated in comparison with existing HAS products. In this section these systems are examined in terms of their design, what they guard against, and what is required to keep them running and to provide confidence that they will work correctly at failover. The ATLAS architecture is then examined in this context in Section 5.

Current HAS products fall into three categories, examined in the following sections.

1. Hot-swap servers.
2. Non-stop computers.
3. High Availability (HA) clusters.

4.1 Hot-swap Servers

4.1.1 Design and Function

The least expensive and simplest approach to an HAS is what we refer to as a “hot swap server.” Such servers address the possible failure of three system components (disks, power supply, and fans) by providing hot-swap RAID disks with redundant hot-swap power supplies and fans.

4.1.2 Constraints

Hot-swap servers do not address the other significant possible failures such as motherboard, CPUs, memory, I/O buses, internal cabling, disk controllers, or network interfaces. Since problems with these components can be intermittent and time consuming to diagnose and repair, substantial downtime can occur. Thus, they are not a true HAS.

4.2 Non-stop Computers

4.2.1 Design and Function

At the other extreme, non-stop computers provide continuous, non-stop computing across all hardware failures. In these computers (which have been around for decades), each hardware component is replicated and hot swappable, including CPUs, motherboards, controllers, and RAID disks. All operations are carried out in parallel on all components and the results are compared. A failing component is automatically detected and disabled, the user is notified, and it can then be hot swapped. One has confidence that the system will work correctly when a component fails since all components are running all of the time. For example, if a replacement component does not work correctly with the existing O.S. (e.g., if it requires a patch or new driver), or if it exhibits problems in the way a service uses it, these will be immediately detected.

4.2.2 Constraints

Non-stop computers are expensive, and generally not cost effective for small businesses. This is especially true if more than one server is required to run the essential services, as is often the case.

Part of the high cost is couched in the following. Non-stop computers can only work if all replaced components are fully compatible with all existing components, in terms of both hardware and software. This is difficult in today's rapidly changing technology environment. For example, 2 years from now, can one find a replacement motherboard that will run correctly with the operating system and all applications currently running on the older motherboard? Or, if patches or new drivers are required for a replacement component, will that updated software also run correctly on the older preexisting hardware? Considering that newer non-stop computers must continue to utilize newer technology, the need to maintain compatible replacement technology for all older units is certainly a costly constraint.

4.3 HA Clusters: Design and Function

HA clusters have always been an alternative to non-stop computers. These consist of two or more computers, either at the same location sharing a common RAID disk array, or at geographically different locations connected by a network. In either case, if one computer fails, another computer automatically takes over and runs the services previously running on the failed computer. We refer to this automated process as “auto-failover.”

4.3.1 Distributed HA Clusters

Distributed HA clusters utilize computers at different geographic locations, with software that maintains synchronization, ensures consistent data for all clients, and initiates auto-failover as required. These insure against not only hardware component failures, but also entire site failures (e.g., due to fire), and can provide for geographic load balancing. Distributed clusters are expensive to purchase and require substantial IT resources to integrate and maintain, and as such are rarely applicable for small business. They are mentioned here only for completeness, and will not be considered further in this paper.

4.3.2 HA Clusters with Shared Disk Arrays

The most common and least expensive HA clusters consist of two adjacent computers, a primary and a secondary, which share a common RAID disk array. The secondary senses the operating status of the primary by periodically sending a message via the network or other communication channels, and expecting a reply. This is commonly referred to as “heartbeat ping.” If the secondary believes that the primary has failed, it initiates auto-failover. It then takes over the role of the primary, starts up all of the primary services, and begins to access the filesystem that the original primary had been accessing (via the shared disk). This type of HA cluster is the only real option for small business, and is what the remainder of this paper will refer to.

4.4 HA Clusters: Constraints

4.4.1 Danger of Split Brain and Complexity

It is essential that both computers never access the same filesystem at the same time, since this would likely cause the loss of data on that filesystem. This is because each computer could overwrite the data from the other, possibly corrupting the entire directory structure. No level of RAID can prevent this. This undesirable situation can occur if both computers believe they are the primary, and is commonly called "split brain." For example, if the primary temporarily hangs and cannot respond to the secondary heartbeat pings, the secondary will initiate auto-failover, but may be unable to ensure that the primary is actually shut down. If the primary later recovers, split brain can occur.

For this reason, the more comprehensive HA clusters will incorporate multiple communication channels, and background watchdog processes in addition to the normal heartbeat ping daemons, in an attempt to prevent this. Although essential to ensure safety, this software and hardware does add additional mechanisms that must be managed, and increases the number of variables that can cause problems at failover.

4.4.2 Lack of Confidence at Failover

As mentioned earlier, a non-stop computer provides high confidence that it will work correctly at failover, by nature of its design. In contrast, an HA cluster provides little confidence that it will work correctly at failover, also by nature of its design. This is because the primary and secondary computers are completely separate machines, which over time, will likely end up with different

combinations of different versions of hardware, firmware, device drivers, patches, application software, or even operating systems. Even seemingly minor differences can cause programs to run differently. Whether the services will run correctly on the secondary at failover, if they will even run at all, depends entirely on the compatibility of these two environments.

The installation of two computers with identical hardware, software, and configurations, is simple. However, it becomes increasingly difficult, and expensive, to keep those two environments from diverging over time as routine changes and repairs are made.

Some changes merely must be made to both computers in order to ensure compatibility. However, it is easy to forget to make the changes in the second computer, or to make them incorrectly, given a typical hectic business environment. Examples include:

- Installing security patches or bug patches.
- Modifying the TCP/IP configuration (IP address, subnet mask, etc.)
- Adding new desktop client addresses to the access control rules.

Some changes, on the other hand, cannot necessarily be made to both computers. Doing so may not be cost effective, or may require undesirable downtime or off-hours service. Examples of these include:

- Replacing a failed network card, which requires a new driver and possibly one or more OS patches. This may also invalidate license keys based on Ethernet address.
- Replacing a RAID controller, which may also require a new driver or OS patches. This must be fully compatible with the RAID controller on the other computer, since at failover, it must access the filesystem written by that other card. Furthermore, the other card must eventually be able to read what the new card has written.
- Replacing an old motherboard with a newer version, since the older one is no longer available. This may require a newer version of the OS, not just a patch. This newer OS may employ different configuration tools (such as Red Hat ipchains vs. iptables), thus making even simple configuration changes more cumbersome to keep in sync. This new hardware may also invalidate license keys.

The fundamental issue is whether the software services will run correctly in the new environment. Underlying this are the lower level compatibility issues, such as those pointed out with the RAID controller and motherboard. Of course, this all could be addressed by always making the same changes in both computers, but this increases the hardware cost and requires even more IT resources, along with downtime or off-hour service.

The types of changes and reasons for them are endless, but they are guaranteed to occur, and the primary and secondary environments will diverge over time. The more they diverge, the more likely it is that the secondary will not run the services the same as the primary, if it will even run them at all.

Consequently, unless mechanisms exist that (1) attempt to maintain compatibility between the primary and secondary, and (2) continuously verify that services will run correctly on the secondary at failover, there is no reason to have confidence in the HA cluster at failover.

Current HA clusters do not provide such mechanisms. The end user is left to their own resources to provide them. This generally implies periodic reevaluation and testing during off-hours. However, the end user may be unable to implement such meticulous ongoing procedures due to limited IT resources.

4.4.3 Load Sharing: Asset or Liability?

HA cluster literature may mention the ability to distribute services across both computers during normal operation, thus not leaving the secondary idle and wasting that resource. If either computer fails, the other will run the failed computers services along with the services it was already running.

This can and does work. However, this increases the complexity of the setup and configuration. More importantly, this exacerbates the already difficult problem of ensuring correct operation at failover. For example, at failover, all services must now run simultaneously with other services that they did not run with before. How can one know that the services will not interfere with each other in unpredictable ways? As another example, a particular service may be expected to exist at the same IP address after failover. If the backup computer switches to that IP address at failover, this may cause a problem for other services already running on the backup computer. Of course, multiple network cards and multi-homing can solve this, but again with added complexity.

This added complexity comes with two costs. Since these costs can be substantial, it is important to weigh them against the added benefit of load sharing during normal operation:

1. The cost of IT support, especially for the more complex ongoing testing.
2. The cost incurred if the cluster does not work correctly at failover.

4.5 Summary of Existing High Availability Servers

Hot-swap servers, although inexpensive, are not a true HAS. Non-stop computers require minimal ongoing IT support but are too costly for most small businesses, especially if multiple servers are required. Distributed HA clusters are expensive to purchase and require substantial ongoing IT support, and are rarely cost-effective for small business.

Thus, HA clusters using shared disk arrays have been the only viable option for small business. They are much less expensive than non-stop computers to purchase, provide automated failover, and provide the possibility of load sharing. However, they require substantial ongoing IT resources in order to ensure that they will work correctly at failover.

5 The ATLAS Architecture

This section discusses the ATLAS design goals, architecture, and the rationale behind the architecture. It also compares ATLAS with existing HAS products in terms of failure modes, required ongoing IT support, and level of confidence at failover. Following is a brief overview, with each issue examined in the following sections.

ATLAS is designed to achieve three essential requirements:

1. Minimize cost.
2. Minimize required IT support.
3. Maximize confidence that the system will work correctly at failover.

These requirements are met through two fundamental design criteria:

1. Remove all unnecessary complexity.
2. Provide automated tools to ensure correct operation at failover.

By removing unnecessary complexity through a manual-failover approach and reduced complexity architecture, the system has fewer variables and fewer aspects to configure, maintain, and test over time. This reduces the required ongoing IT support. Further, this increases confidence at failover since there are fewer things that can go wrong.

Then, instead of this *unnecessary* complexity, we provide the *necessary* mechanisms to automatically ensure that the system will work correctly at failover. This relieves the end user of this substantial, continuing burden, which exists with other HA clusters. The result is a system in which one can have confidence, with minimal ongoing support.

5.1 Minimizing Complexity

The ATLAS reduced complexity architecture with manual-failover strikes an important balance between no failover capacity, and HA clusters with auto-failover that (1) can cause more problems than they attempt to prevent, and (2) require substantial IT resources to manage. Following are some examples of how the ATLAS architecture decreases the amount of IT support required, while increasing confidence at failover.

5.1.1 No Danger of Split Brain

As discussed in Section 4.4.1, a fundamental issue with HA clusters is "split brain", where both servers mistakenly become the primary, which can lead to loss of data. With our manual failover approach, such data loss simply cannot occur since there are no shared disks. Further, both servers can never think they are primary since the “role” of the server (primary or secondary) is defined by the swappable disks, and they can be in only one server at a time. This design obviates the need for the additional software complexity to (1) handle the auto-failover, and (2) guard against split brain. The end result is fewer mechanisms that must be managed and tested, and fewer variables that can cause problems at failover.

5.1.2 Increased Confidence through Manual Failover

In Section 4.4.2, the critical issue of confidence at failover was examined. To this end, our manual failover approach requires that both servers be shutdown and restarted. This ensures that, at failover, all services start in the same manner and with the same system state as they did originally, reducing the likelihood of transient problems and increasing confidence at failover -- all without requiring any complex and special software.

5.1.3 Minimizing Cost by not Load Sharing

As discussed in Section 4.4.3, clusters provide an opportunity to distribute services across the computers during normal operation. However, doing so results in increased IT support costs, especially for the more elaborate ongoing testing that is required, while also increasing the possibility of problems at failover. For these reasons, ATLAS does not allow load sharing. During normal operation, all services run on the primary only. The secondary is dedicated to the tasks involved with ensuring correct operation at failover and monitoring the primary services (as discussed in Section 5.2). This approach reduces the required amount of ongoing IT support, while simultaneously providing a higher level of confidence at failover. This essentially trades the known upfront cost of the secondary server for the unknown future costs of support, and of the system not working correctly at failover.

5.2 Maximizing Confidence at Failover

As discussed in Section 4.4.2, with any HA cluster it is essential to provide some means of ensuring that it will work correctly at failover. ATLAS ensures correct operation at failover by automating the following three tasks, which are examined in next sections.

1. Maintain compatibility between the primary and secondary servers.
2. Verify that all services run correctly in the failover environment.
3. Provide a continuous status report.

5.2.1 Maintaining Primary and Secondary Compatibility

In Section 4.4.2, we examined the importance of maintaining compatibility between the primary and secondary servers to ensure correct operation at failover. Although it is simple to initially set up two servers with identical hardware and software, the systems will undoubtedly diverge over time, as changes are required in hardware, software, and configurations. The more the two servers diverge, the more likely it is that the secondary will not run the services the same as the primary, if the services will even run at all.

Current HA clusters provide no mechanisms to address this issue. The end user must design and implement procedures to maintain compatibility.

With ATLAS, automatic mechanisms that ensure compatibility have been designed in from the start. ATLAS automatically maintains server compatibility in the face of changes to hardware, software, and configurations. These mechanisms are now described.

The ATLAS Sentry Synchronizer is responsible for maintaining compatibility between the primary and secondary environments. To meet this responsibility, the Synchronizer performs the following three tasks for each supported service, utilizing the existing business network. These are each discussed in the following paragraphs.

1. Ensure that primary and secondary configurations are tracking, even across different configuration versions and formats.
2. Ensure that verification test data correctly exists on the secondary.
3. Manage all changes in a version control system.

First, the Synchronizer ensures that the configurations on the secondary correctly reflect the meaning of the configurations on the primary. *This is not simply a matter of copying the files across.* Configuration files can change with different versions of software. Some changes are simple, such as a new file location or name. Some changes are more involved, such as altering the syntax or semantics of the file contents. An example of this would be a change from non-shadow passwords to shadow passwords. However, some changes are more complex, such as Red Hat Linux changing the method of network access control from ipchains to iptables.

Regardless of the type of change, the Synchronizer handles the differences by automatically converting from the version on the primary to the version required on the secondary. If the differences cannot be handled correctly, perhaps due to incompatible software versions or malformed configuration settings, the Synchronizer will report this. The problem can then be resolved before a failover event occurs.

Second, the Synchronizer must ensure that the test data, which is required to verify each service (discussed in Section 5.2.2), exists on the secondary and correctly reflects the contents of that test data on the primary. As with the configuration files, software version differences between the primary and secondary are automatically handled, and any problems are reported. Note, any such problems would probably also be reported by the Verifier (Section 5.2.2), since the secondary services would likely not perform as expected.

Finally, all changes to all configuration or test data files are managed by a version control system. This allows one to view each change, when it was made, and to roll back to a previous state if necessary. This occurs regardless of how the change was made. For example, the user authentication files might be modified in a text editor, or updated automatically, either from an account wizard GUI or as a side effect of installing software that required a new user account. Regardless of how it was done, the change will be recorded in the version control system, and the secondary authentication files will be updated accordingly.

5.2.2 Verifying the Failover Environment

Maintaining compatibility is just the first step to ensuring correct operation at failover. As discussed in Section 4.4.2, unless mechanisms exist which continuously verify that services will run correctly on the secondary, one can have little confidence in the HA cluster at failover.

Current HA clusters provide no mechanisms to address this issue. The end user must design and implement procedures to ensure correct operation at failover.

With ATLAS, automatic mechanisms to verify the failover environment have been designed in from the start. ATLAS continuously runs, exercises, and verifies all services on the secondary, automatically reporting any problems. These mechanisms are now described.

It is important to understand that the secondary automatically runs all services exactly as the primary does. This is due to the Sentry Synchronizer, which ensures that the secondary configurations are tracking with the primary configurations. Consequently, the secondary takes exactly the same actions as the primary at boot time.

The ATLAS Sentry Verifier is responsible for continuously verifying that all services run correctly on the secondary. To meet this responsibility, the Verifier remotely exercises each service, utilizing the existing business network, and verifies the results. Anything that would cause a service to run incorrectly on the secondary will be detected and reported, regardless of the cause. It will detect not only the obvious issues such hardware or network faults, but also software incompatibilities, configuration problems, or even a malfunctioning Synchronizer. The problem can then be resolved before a failover event occurs.

The one difference between services running on the primary and secondary is that they access different RAID disks. Although the secondary RAID disks will not contain all of the application data that exists on the primary RAID disks, the Synchronizer ensures that the secondary RAID disks do contain the configurations and data that are required to run and verify each service.

For example, to exercise and verify a web server the Verifier may perform the following tasks:

1. Open a static web page and verify the contents returned.
2. Open a web page that is dynamically created from a database, and verify the contents returned.
3. Open a web page that requires Java applets, and verify their operation.
4. Verify that restricted web pages are, in fact, restricted. And so on.

The file structure and data required to support these tests will be installed on the primary as part of the Verifier setup. This would include the static and dynamic web page files, the necessary CGI programs, the test database, and so on. The Synchronizer then ensures that this file structure and data will also exist on the secondary. If the verification procedure is later modified and the necessary data changes, the Synchronizer ensures that these changes are propagated to the secondary. Thus, the web server runs on the primary and secondary with identical configurations, and can be verified on the secondary even though it does not have access to all of the application data on the primary RAID disks.
Since the ATLAS architecture is symmetric, not only does the primary Verifier verify the secondary services, but the secondary Verifier also verifies the primary services as well. Thus, ATLAS not only ensures correct operation at

failover, but also monitors services on the primary during normal operation, all with the same software.

Each Verifier also monitors its local servers overall health, and other entities on the network such as client workstations, DNS servers, or routers. The collective monitoring results from both Verifiers are a valuable aid in determining the cause of problems, which can help assess whether a problem lies in the network, the server, a particular service, or a particular client.

5.2.3 Status Reporting

All status information from both the Synchronizer and the Verifier is reported through a monitoring mechanism on each ATLAS server. The output can be viewed either from a web browser or directly on the server consoles. Status alerts and problems can be reported with selectable visual or audible effects, as well as through email, pagers, etc.

5.3 Summary of the ATLAS Architecture

5.3.1 Maximize Confidence / Minimize Cost

ATLAS is designed to maximize confidence at failover, while minimizing both the upfront cost and the ongoing IT support cost. These goals are achieved by (1) utilizing a reduced complexity architecture, augmented with (2) an automated mechanism to ensure correct operation at failover.

5.3.2 Reduced Complexity Architecture

The reduced complexity architecture includes characteristics such as manual-failover (instead of auto-failover), the requirement for a full reboot at failover, and not allowing load sharing. As a result, ATLAS has fewer aspects to configure, maintain, and test over time, which reduces the amount of ongoing IT support that is required. These also increase confidence at failover, since there are simply fewer things, and combinations of things, that can go wrong.

5.3.3 Providing Confidence at Failover

The automated mechanism to ensure correct operation at failover is provided by the ATLAS Sentry software suite. The Sentry Synchronizer ensures that the primary and secondary environments remain compatible across configuration changes as well as different versions of hardware and software. The Sentry Verifier ensures that all services will run correctly at failover by continuously exercising and verifying the operation of each service on the secondary, as well as on the primary. Any problems from either the Synchronizer or the Verifier are automatically reported through a flexible monitoring system, and can be remedied before a failover is required.

By automatically ensuring correct operation at failover, ATLAS relieves the end user of this substantial, endless IT support burden. The result is a considerable reduction in the amount of ongoing IT support that is required, while providing an HAS in which one can have confidence.

6 ATLAS Specifications

6.1 Applicability

- The design of ATLAS is operating system independent. It can be implemented for any OS and associated services.

- Currently, ATLAS runs on Linux, and supports Linux services such as:
  - Samba server (with Active Directory)
  - NFS server
  - Mail (SMTP) server, either postfix or sendmail
  - Apache Web server with PHP extensions
  - OpenLDAP directory server
  - MySQL or Postgres SQL database servers
  - Kerberos security

- Support for Windows and Windows services is planned for the future.

- ATLAS manual failover requires a person on-site to shutdown both servers, swap the front mount disks, and reboot the servers.

6.2 Features

- Dual redundant high-performance commodity servers.
- Servers can be configured as required (CPU, memory, disk, etc.).
- Dual redundant uninterruptible power supplies (UPS).
- RAID Level 1 mirroring of all data.
- All disks are front-mounted in swappable trays.
- Commodity hardware support can be provided by anyone.
- Continuous tracking and syncing of server environments.
- Continuous exercising and verification of services on both servers.
- Continuous monitoring of overall health of both servers.
- Continuous monitoring of the network.
- Real-time display of monitoring results via web browser.
- No need to modify applications in any way.
- Applications need not be "cluster aware."
- No danger of data loss from a malfunctioning auto-failover.
- Optional RAID Level 5 or 10 (1+0).
- Optional tape backup system.
- Optional rack unit (everything contained in a compact 8U open rack).
- Optional tri-server system to ensure against 2 hardware failures.