Chainsaw to Assured Availability

Urbiš Miloslav
urbis@scova.vabo.cz

VÚ8918/05
Bystrice p. H.

Abstract

This document describes the design and implementation of high availability systems and networks. Uninterrupted access to applications and data is the most important goal of today. This paper is about high availability (HA) computing through IT infrastructure. The central goal of this paper is the description of the difference between traditional solutions (clusters, Storage Area Network etc.) and new technologies.

Keywords: high availability, redundancy, clusters.

1. Introduction

This document will describe basic aspects of building 99.999% IT infrastructure. In particular, I want to provide readers with the following themes:

- Typical Single points of Failure,
- High Availability Clusters,
- Redundant network infrastructure,
- Solutions for Assured Availability.

2. Typical Single Points of Failure

Given a system to be designed, the final issue is to provide the system and security administrators with security policy specifications, in order to protect the system efficiently and sufficiently against any threat of its security properties (confidentiality, integrity and availability). Not surprisingly, several successive steps are necessary for the security methodology:

- The first one consists of a risk analysis: which are the threats we want the system to be protected against? What are their characteristics?
- Given a set of threats, the second step is to specify a security policy to combat them. We can use security criteria ITSEC, Common Criteria etc. to ensure confidentiality. There are not practical criteria for availability. This problem must be solved by a good analyze and methodology of design.

Typical failures (threats) of IT components are shown in the following table, together with a description of how the single point of failure (SPOF) can be eliminated.

Component	**What Happens if Component Fails**	**How the SPOF is Eliminated**
Single SPU	Service is lost until the SPU is repaired	Provide a backup SPU to the host application (cluster of host systems).
Single LAN	Client connectivity is lost	Install redundant LAN interface cards and subnets. Configure stand-alone interfaces.
Single LAN interface	Client connectivity is lost	Install redundant LAN interface cards, or configure standby LAN interfaces in a grouped net.
Single root disk	Service is lost until disk is replaced.	Use mirrored root disk.
Single Data DISK	Data is lost	Use mirrored storage for individual disks or use disk arrays in data protection mode.
Single Power Source	Service is lost until power is restored.	Use additional power sources, and employ UPS technology on each.
Single Disk Interface Card	Service is lost until card is replaced	Dual or redundant SCSI cards with dual I/O path to a disk array.
Operating Systems	Service is lost until OS reboots	Provide failover capability, and tailor applications to restart and recover.
Application Program	Service is lost until application restarts	Provide a facility to restart the application automatically. Tailor applications to restart and recover. Provide careful, thorough debugging of code.
Human Being	Service is lost until human error is corrected	Automate as much operation as possible. Document procedures thoroughly.

Table 1. Threats and Eliminating Single Points of Failure

There are many ways, how to eliminate SPOF. This problem is discussed in the following section.

3. High Availability Clusters

3.1 Basic Concepts

Several approaches to building fault tolerant and high availability systems exist; however, certain characteristics are common to any system that is built with less than perfect components (which, of course, means all components). Namely, in order to manage a failure, there must be an alternative component that continues to work properly. Thus, redundancy is a fundamental prerequisite for a system that either recovers from or masks failures. Redundancy can be provided in two very different ways: *passive redundancy* and *active redundancy*, each of them with very different consequences.

A *passively redundant* system provides access to alternative components that are not associated with the current task and must be either activated or modified in some way to pick up the load of the failed component. The consequent transition is noticeable and may even cause a significant interruption of service. Subsequent system performance may also be degraded. Examples of passively redundant systems include standby servers and clustered systems. The mechanism for handling failures in passively redundant systems is to *fail over* to an alternative server. The current state of the failed application will be lost, and the application must be restarted in the other system. The failover and restart processes typically cause some interruption or delay in service to the users. Thus, passively redundant systems, such as standby servers and clusters, provide high availability but cannot deliver continuous service and assured availability.

An *actively redundant* system provides at least one alternative processor that runs concurrently on the same task and, in the presence of a failure, provides continuous service without a noticeable interruption in service. The mechanism for handling failures is to compute through a failure on the remaining system or systems. Because there are at least two processors looking at and manipulating the same data at the same time, the failure of any single component will be invisible to both the application and the user.

Most solutions offered today, whether passively redundant systems from Legato/Vinca, Microsoft, IBM, Compaq and others, or actively redundant systems from Stratus or Tandem/Compaq, require that the operating system/drivers and/or the application have specific knowledge of the system architecture in order to take advantage of the available redundancy. This knowledge is imparted to the operating system/drivers and applications by creating or modifying them to conform to a specific Application Programming Interface (API), a costly endeavor. Thus, operating systems/drivers and applications written for a standalone system cannot easily take advantage of the extra reliability available when the software is loaded onto one of these redundant configurations unless they are modified specifically for that system.

Now that we understand the fundamentals, let's examine some systems - from the simple two power supply system to assured availability and disaster tolerant systems.

3.2 Recovery Based Systems

Recovery based systems that use passive redundancy fall into three categories: backup, standby servers, and clusters. Our discussion here focuses on the more feature rich systems represented by clusters.

3.2.1 Clusters

There are several different configurations for these kinds of systems; the first uses two complete systems connected by some kind of network connection, usually a high speed Ethernet, as shown in Figure 1. Here data stored on the systems are mirrored over this path and can be found in the Vinca Cluster as well as the Octopus backup system. This path also carries messages from each system concerning their mutual status "heartbeat". messages. The user data is mirrored over the private network so access is assured for all users should one server fail.

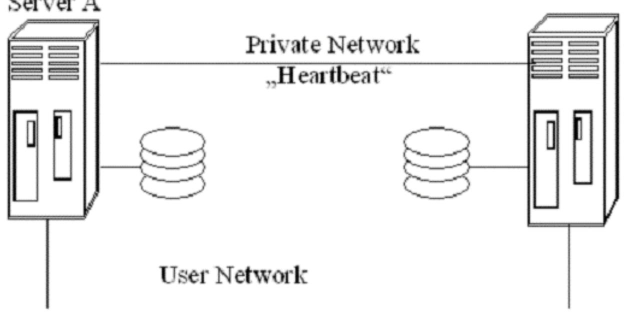

Figure 1. Two Node Cluster Configuration (Vinca Type)

The other configuration, shown in Figure 2, is associated with Microsoft's Cluster Server. The storage system is divided into two parts, with each part dedicated to one of the servers. Common access is provided by a SCSI bus connected between two systems, with the disk drives for both systems connected to the bus. In normal operation, each server accesses its own drives, the two systems operate independently, and the private network connection carries ”heartbeat” messages between the two systems. If one system should fail, the users can be logged onto the surviving system and their applications restarted. All the disk drives connected to the common SCSI bus are then available to the surviving system.

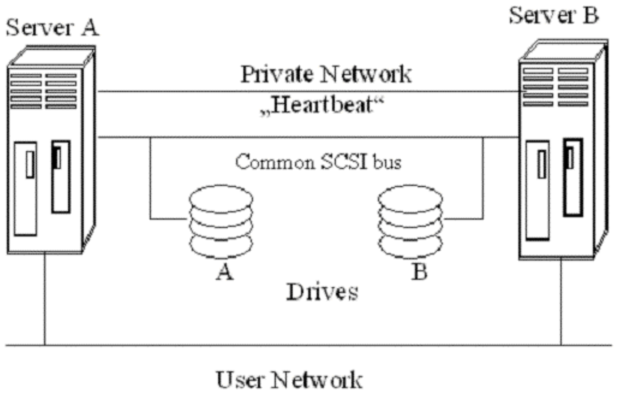

Figure 2. Simple Cluster Configuration (MS Cluster Server Type)

The repair and recovery cycle for simple clusters requires that once the failed system has been repaired, both systems must be rebooted. The reboot is necessary because the drives must be reallocated across the SCSI bus, and users reassigned to the two processors.

Thus, simple cluster architectures provide a failover mechanism that allows one processor to take over for another in the event of a failure. Cluster systems do not provide assured availability in which failures, and repair/recovery cycles are transparent to the users, nor do they provide continuous processing.

Another cluster configuration uses a shared data architecture, which gives multiple computers access to the same disks. The sharing features are facilitated by two key components: a fault tolerant file system with universal access, and a distributed lock manager for controlling access to and modification of common files. Shared access clusters were created in the early 1980s by Digital as a way to deliver improved performance as well as availability for its uniprocessor VAX/VMS computers, as shown in Figure 3.

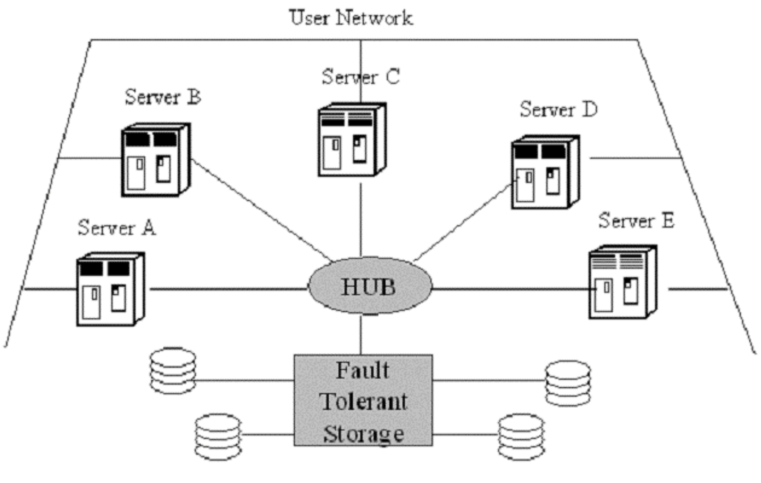

Figure 3. Shared Files Cluster Configuration (Digital VAX Cluster Type)

Each server is connected through a redundant high-speed hub or switch to each other and to the fault tolerant storage system. The fault tolerant storage system provides high-speed access, disk caching and mirroring, as well as handling backup to archival storage. Note that because the storage system is independent of the servers, the repair/recovery process does not require rebooting the whole system and the concomitant interruption of the users. The VAX cluster type configuration is available only from Digital/Compaq and is not available on the Intel architecture and Windows Server operating system.

Clusters as well as standby servers can restart applications that have been properly programmed for cluster operation on an alternate server; however, the state of the application before the failure occurred will be lost. For stateless applications, such as a digital clock that merely reads the time from the real-time clock in the server and displays it on the monitor, clusters are usually an acceptable solution. Restarting such an application on an alternate server will simply display the current time. A simple application with a state is the solitaire game. If a server fails in the middle of a game, the alternate server will restart a new game, and the state of the game on the failed server will be lost forever. Therefore, some clusters such as the Oracle Parallel Server (OPS) deliver scaleable performance for properly written applications as processors are added. All clusters provide high availability for properly written applications that can be restarted on an alternate server.

4. The Redundant Network Infrastructure with Fault - Tolerant Solutions

4.1 DIHI

The Distributed Internet High-availability Infrastructure (DIHI) provides a scalable and highly available architecture for Internet applications. DIHI employs load balancers for the dual purpose of availability and scalability enhancement. For example, if load balancers are used to scale Web-serving capabilities and one Web server goes down, there are still several others available to process the user's Web requests. Figure 1 shows an example of DIHI architecture with one firewall. Obviously, if the firewall fails for any reason, communications between a company and the Internet will be terminated and cannot be restarted until the firewall is operational. The succeeding sections will describe implementations of a HA solution for firewalls to avoid this potentially critical problem.

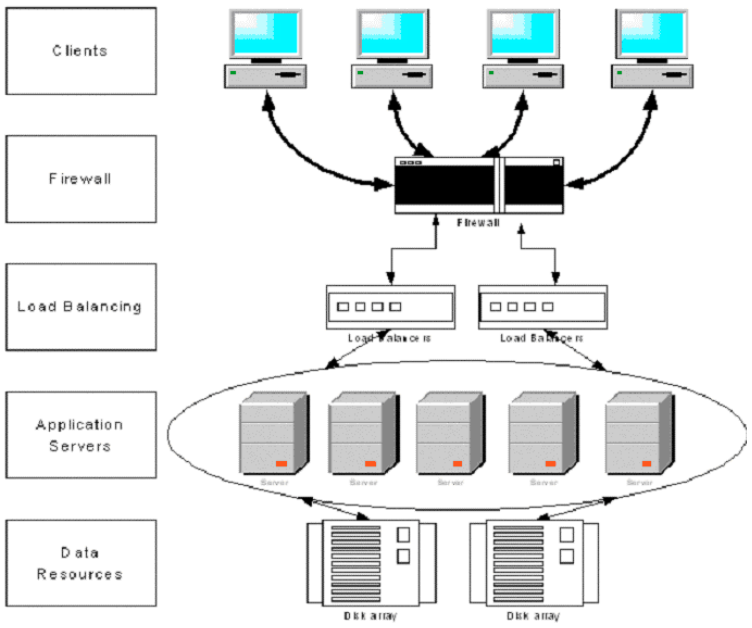

Figure 4. The Distributed Internet High-availability Infrastructure

As part of DIHI architecture, it is essential to have a highly available firewall, which can be accomplished with a combination of hardware, operating system, and application software redundancy. This document has described two solutions to accomplish HA for firewalls:

- Check Point FireWall-1
- AXENT Raptor Firewall

4.2 Firewalls and DIHI

Firewalls are the best practice for portal protection when private networks are connected to the Internet. Firewalls have traditionally been implemented without any means of redundancy, with the possible exception of a standby server that must be switched over manually to assume the role of the primary server. This manual switchover can lead to several hours of downtime. The following sections describe the products and these two approaches taken to obtain high availability:

- Using load balancers.
- Using an application-clustering solution.

Providing HA for firewalls is, in general, more difficult than providing HA features for application servers (such as Web servers). Due to the nature of the function that firewalls perform, it is essential that the firewall maintain a detailed track of the ”state” of each network connection. How firewalls keep track of ”state” information, and how much information they retain, it varies between those firewalls that implement stateful inspection and those that use application proxies.

The necessity to maintain state information has made it difficult in the past for a session through a firewall to continue uninterrupted, if the firewall becomes non-operational and is ”failed over” to another one. The unique nature of the function and implementation of firewalls is the reason that off-the-shelf clustering and load balancing software from OS manufacturers may work adequately with some applications, but not with firewalls. The amount of data ”lost” during a fail-over event will vary, depending on the implementation approach.

Note: State information refers to the firewall’s ability to keep track of the status of all connections with and through the firewall. TCP connections have more state information associated with them than UDP connections.

Firewall vendors have taken different approaches to providing state information that can be shared among firewalls for the purpose of fail-over. Some vendors provide as much state information as possible. Other vendors intentionally provide less state information. These vendors view fail-over events as security risks. Therefore, to minimize this risk, their approach is to revalidate all connections to ensure that the initial reason for the firewall failure was not due to a malicious attack, which could be carried over onto the backup firewall if all state information was retained.

4.2.1 Stonesoft StoneBeat 3.0

Stonesoft StoneBeat 3.0 is a software solution that enables the building of a continuously available firewall system by using Check Point FireWall-1 and Windows NT. This combination of products allows companies to build business-critical firewall systems that are highly available.

StoneBeat protects the firewall system against hardware and software failures by allowing for a hot standby configuration, enabling fail-over in the event of a firewall host failure. Hot standby, as the name implies, is a configuration in which one server is actively running the FireWall-1 software and processing network traffic. The backup server is also running the FireWall-1 software, but is processing no network traffic through the firewall. In the event of a hardware or software failure on the primary server, a fail-over occurs to the backup server. Thus, minimal session information is lost during the fail-over.

In addition to hot standby, load sharing is another fail-over technology that allows the use of multiple firewalls configured similarly with all of them operational, thereby sharing the load as it comes into the network. Load sharing differs from load balancing in that the decisions on what traffic goes to which server is made by setting parameters ahead of time, rather than routing traffic dynamically as with a load balancer. Load sharing may require a network to be split, depending on the requirements of the load sharing policy. In addition, load sharing can be accomplished across multiple Stonebeat clusters.

Firewall Throughput	**HA Solutions**
Less than 20Mb/s	one firewall nodes
Less than 50 Mb/s	two firewall nodes in a hot standby configuration
Less than 100 Mb/s	two unit in load sharing configuration
More than 100Mb/s	three units with load sharing and clustering

Table 2. Firewall Throughput vs. HA Solutions

StoneBeat also allows manual fail-over for maintenance operations during normal business hours without interrupting the firewall operation. The StoneBeat software modules (with the exception of the management control console software) are loaded on the same servers as the firewall and can detect hardware failures in the following two ways in the hot standby configuration:

- The two systems are connected to each other using a serial cable that monitors a ”heartbeat;”
- the absence of the heartbeat will trigger a fail-over.

A test subsystem included with StoneBeat can be configured to detect several types of failures, such as:

- common hardware failures (including fixed disk driver or network interface problems),
- fail-over software failures.

The test system can also be modified and configured to perform the following tasks:

- customize tests, such as running a customer-created executable program and then verifying the expected result,
- monitor the operating system for resource starvation,
- poll the firewall service status,
- check the installed security policy.

The StoneBeat application is managed using a Windows NT GUI (see Figure 5) or command line utility on the Control server. Using these utilities, an administrator can switch an active firewall offline and perform maintenance tasks while the hot standby system performs the firewall tasks. Normally, for firewalls at one site, the Check Point FireWall-1 Management Console would be installed on the same server as the Stone-Beat software. For multiple site firewall implementations, the Check Point Management Console may be on a separate server, depending on operational needs.

When the system makes a switch from the primary to the secondary, or vice versa, there is a chance that users will lose some existing connections. The result depends on the following parameters:

- FireWall-1 synchronization;
- TCP/IP protocol of each connection;
- FireWall-1 address translation rules in use;
- FireWall-1 VPN and SecuRemote usage;
- FireWall-1 version.

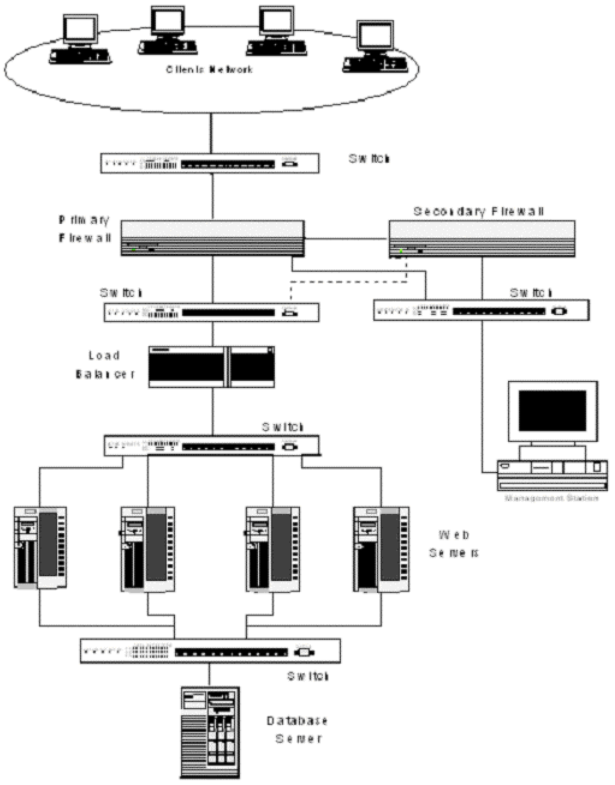

Figure 5. Stand-by solution with Firewall-1

Synchronization

FireWall-1 synchronization is a very important feature and existing connections need to be preserved. Synchronization should be enabled in both directions between primary and secondary. If synchronization is not enabled, you will lose some of the connections during switchover. Examples of lost connections are those based on the FTP protocol. In FireWall-1, the FTP data connections are passed through the firewall based on the state tables. See the Check Point FireWall-1 Architecture and Administration Guide for details. Simple TCP/IP based connections, like Telnet and HTTP, may be preserved in a switchover even without FireWall-1 synchronization, depending on the direction of the first packet after the switchover. If you are using the FASTPATH option of FireWall-1, these connections are preserved in the switchover.

NAT

When using Network Address Translation (NAT), simple TCP/IP-based connections, like Telnet, are preserved in the switchover. Some other connections, like FTP, need to be re-established afterwards.

FireWall-1 VPN

FireWall-1 VPN connections are preserved through switchover when some encryption schemes are used (FZW); however, others are not preserved.

Fail Over

Check Point retains state information of its firewalls on the backend management station. Since there is a heartbeat line continually maintained between the two firewalls, the StoneBeat software uses all pertinent state information in re-establishing connections automatically to prevent loss of data. The Stonesoft software took between 15 and 30 seconds to completely fail-over to the backup system. During this time, incoming packets were queued until the firewall was fully functional. No packets were allowed through the firewall until the firewall was functional.

4.2.2 Raptor Firewall and RADWARE FireProof Load Balancers

High Availability was a basic requirement for the firewalls with IP Load Balancer Devices (IPLBD) The RADWARE FireProof system is a dynamic load balancing system for effective management of traffic on multiple firewalls. The FireProof system is based on existing RADWARE technologies and improves firewall performance, while maximizing uptime. Figure 6 shows a basic installation of an AXENT Raptor Firewall and FireProof load-balancing system. It is recommended that a redundant FireProof system be used in this architecture too.

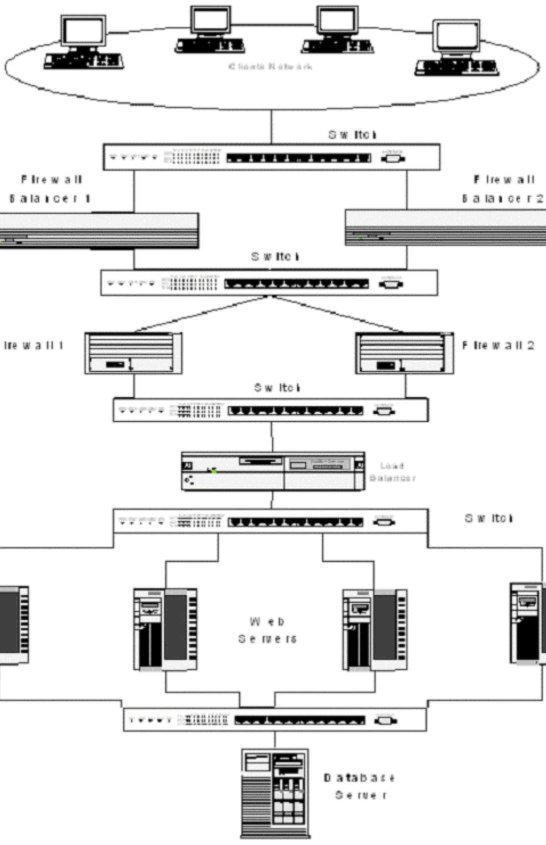

Figure 6. AXENT/Raptor and FireProof Configuration

The RADWARE FireProof system can be configured to work with multiple firewalls, each identically configured. If one of the firewalls were to fail, the FireProof system would no longer send requests to the failed firewall. A failure is determined by polling, a task that the FireProof system accomplishes periodically against each of the firewall servers. If a response is not received in a set amount of time, it presumes the firewall has failed and then the other firewall(s) is/are used. In addition, the FireProof system itself has fault tolerance capabilities. The system can be redundantly configured with two FireProof units. The units check each other's "health" via standard TCP protocols over the network to assure unit and network health. A FireProof unit provides full session tracking and mirrors this state table to the backup FireProof. In addition, some firewall configurations will allow both units to function simultaneously. In the event of a failure, the second unit would take over operations.

The RADWARE FireProof load balancer also works with other firewalls. Each firewall is designed differently so there may be different issues to be addressed, based on the firewall chosen. Network Address Translation (NAT) is a key feature of some firewalls that must be evaluated to ensure adequate functionality when using a load balancing solution.

HTTP requests enter the network at the firewall load-balancing layer. This layer consists of two redundant RADWARE FireProof systems. The FireProof device is configured with a pool of firewalls to which it forwards the incoming requests. Next in the network there are the firewalls: two identically configured servers with Raptor Firewall 6.0 for Windows NT. The firewalls were configured to accept HTTP requests and for redirection. Upon redirection, a proxy connection was established from the firewall to the next layer of the network - the load balancer for the Web servers. The load balancer for the Web server then routed the request to the appropriate Web server. The Web servers, in conjunction with the database server responded to the request and sent back the appropriate content.

5. Solutions for Assured Server Availability

Marathon Assured Availability solutions with patented technology included in the Endurance array product set, provide uninterrupted access to applications and data. Traditional clusters, by definition, cannot do this. Clusters and other standby products are availability solutions that, upon a failure, must stop, restart, reconfigure, and reboot. This can result in considerable downtime, with abrupt service interruption that may lead to catastrophic data loss and data corruption.

Implementing traditional high availability systems involves complex operations and procedures. Every high availability function must be specified, designed, developed, deployed, and maintained at the hardware, operating system, and application levels throughout a system implementation life cycle.

Marathon Assured Availability solutions mask the complexities implementing, deploying, and managing high availability solutions for Windows NT environment. Marathon Assured Availability solutions combine continuous processing and standards-based servers to deliver worry/free Microsoft Windows environments that are easy to implement, quick to deploy, and easy to maintain.

Endurance is hardware and software design to combine four personal computers into a single Windows NT server with no single point of hardware failure. The software applications need not be aware that they are running on a high-availability server. Programs don't need modifications to be fault-tolerant. The failover time is less than it is with software-based failover. Endurance's failover is near zero because the other half of the system is doing the same thing, eliminating any load-balancing possibilities while ensuring that the database files are intact, even if a transaction is executing during the failure.

All I/O task requests from the Compute Element (CE) are redirected to the I/O Processor (IOP) for handling of labor. The IOP runs the Marathon Endurance software as an application, which handles all the fault handling, disk mirroring, system management, and resynchronization tasks. Since Windows Server is a multitasking operating system, other non-fault tolerant applications also can be run on the IOP. This feature allows stateless tasks with fail-over such as Web page services to be run on the IOPs and state sensitive applications such as e-commerce data bases to be run on the CE/IOP combination.

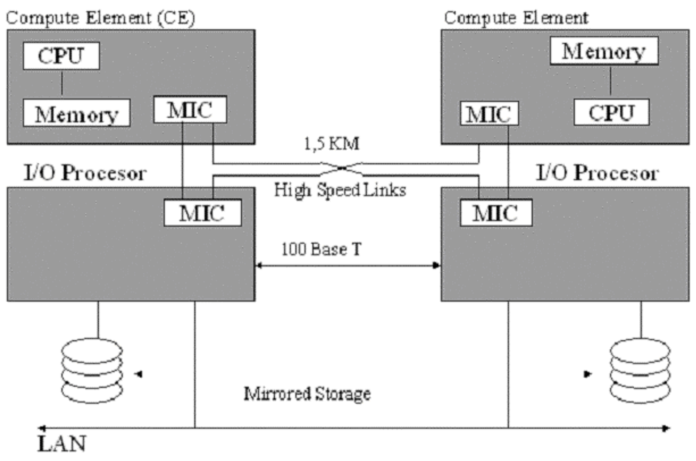

Figure 7. Endurance 4000 Array

The two CEs run Marathon's patented synchronization technology and execute the operating system and the applications in lockstep. Disk mirroring takes place by duplicating writes on the disks on each IOP, thereby providing RAID 1 functionality without a special RAID controller. If one of the CEs should fail, the other CE keeps the system running with only a pause of a few milliseconds to remove the failed CE from the configuration. The failed CE can then be physically removed, repaired, reconnected, and turned on. The CE is then automatically brought back into the configuration by transferring and resynchronizing the state of the running CE to the repaired system over the high-speed links. The states of the operating system and applications are maintained through the few seconds it takes to resynchronize the two CEs, thus minimizing the impact on the users. Note that this is very different from cluster resynchronization, described above. Further, if an IOP fails, the other continues to keep the system running. The failed IOP can then be physically removed, repaired, and turned back on. After the Marathon software starts running, the repaired IOP automatically rejoins the configuration, and the mirrored disks are re-mirrored in background mode over the private Ethernet connected between the IOPs. A failure of one of the mirrored disks is handled through the same process.

The network connections are also fully redundant and work as follows. Network connections from each IOP are booted with the same MAC address, and only one is allowed to transmit messages while both receive messages. In this way, each network connect monitors the other through the private Ethernet. Should either network interconnect fail, it will be detected by the IOP, and the remaining connection will carry the load. The system manager will also be notified of the failure so a repair can be initiated.

Marathon Technologies is, at present, the only supplier to offer 99.999% uptime products that are capable for Windows servers. Next table describes an overview of Traditional Cluster vs. Marathon Endurance Array.

	Traditional Clusters	**Marathon Endurance Array**
Servers	Independent network devices aliased to function as a single system/cluster. Performance can change upon fail over. Single logical server with single points of failure.	Single network device with transparent redundant resources. Constant performance. Single logical server with one IP and one MAC address.
Application	Requires applications to stop and failover. Failures exposed to users. User sessions & context dropped. Application context lost. In/flight transactions lost.	Marathon Computes through failures. Failures masked from users with no application failure. User sessions unaffected. Application context maintained. IN/flight transactions completed.
Data	Access to data interrupted during failover. Mirrored. Data recovery sometimes required.	Continuous data access. Twu simultaneous writes. No loss of in/process data.
Network connectivity	IP aliasing, NetBIOS aliasing.	One IP address and one MAC address transparently connected to multiple resources.
Cluster Scripts & APIs	Cluster script required. Application coding through Clustering API required.	No scripts. No application coding required.
Disaster Tolerance	No. Disaster recovery required.	Yes, with Marathon SplitSite functionality. Separate tuple installations by up to 500 meters.
Administration	Complex – primarily cluster unique.	Standard / Windows and Marathon Manager.

Table 3. Overview of Traditional Cluster vs. Marathon Endurance Array

Marathon Assured Availability solutions combine continuous processing and standards-based servers to deliver worry-free Microsoft Windows environments that are easy to implement, quick to deploy, and easy to maintain.

6. Conclusions

There has been much confusion in the computer industry concerning the concepts of high availability, fault tolerance, and disaster tolerance. The differences should now be clear. A high availability cluster system uses passive redundancy to ”fail over” a user to an alternative system, a process that can take several minutes with loss of the state of the running application. An assured availability fault tolerant system uses active redundancy to ”compute throughle a failure”, a process that is totally invisible to the user, has no single point of failure or repair, and can go through the identification, isolation, repair, and resynchronization steps without losing the state of the user- running applications.

As part of DIHI architecture, it is essential to have a highly available firewall, which can be accomplished with a combination of hardware, operating system, and application software redundancy. This document has described two solutions to accomplish HA for firewalls:

- Check Point FireWall-1.
- AXENT Raptor Firewall.

These solutions employ two different approaches to achieve high availability:

- The Check Point /Stonesoft StoneBeat solution uses cluster software at the application level.
- The AXENT/RADWARE FireProof solution uses load balancer technology in conjunction with the standard firewall software.

Providing HA for firewalls is more difficult than providing HA features for application servers. You can use Marathon Technologies products for Windows NT servers. Marathon Technologies is, at present, the only supplier to offer 99.999% uptime products that are capable for Windows NT/2000 servers.

7. References

[1] Marathon Technologies Corporation.: Assured Availability and High Availability Systems White Paper, Boxborough, 1999.

[2] Marathon Technologies Corporation.: Evaluating Marathon Assured Availability Solutions vs. Traditional Clusters, Boxborough, 2000.

[3] Compaq Computer Corporation: High Availability of Check Point FireWall-1 4.0 and AXENT Raptor Firewall 6.0 for Windows NT in a DISA Environment, 1999.

[4] Radware Corporation.: Technical Application Note 1045 - DNS and the WSD, 1999.