Data Replication for High Availability Web Server Clusters

The price of high performance server hardware has been falling drastically over the last few years. With the advent of open source software and the spread of Linux and server appliances, it is now possible to use relatively low cost, high performance server systems to meet the growing demand of internet and intranet based computing. To effectively use these lower cost, higher performance server systems, solutions for building clusters of servers and server appliances must now be realized. In the past, scaling computing performance has required that lower cost, lower performance systems be replaced with higher end cluster based systems. These systems have used high performance, tightly coupled CPU’s and high cost shared storage systems. Even today, Storage Area Networks (SANs) to allow shared storage for clustered network servers have been all the rage. But what if lower cost, higher performance systems could be clustered and use non-shared storage (basically their existing internal disks) to provide clustered solutions for specific applications such as web servers and web server appliances? How might these high availability clusters be built? What issues need to be resolved to address the requirements for systems that appear to be a single system to the client but in fact are separately clustered servers without any shared components except a high speed LAN?

This paper specifically addresses how to replicate web server data for a variety of different operating systems. A number of alternative sources of information exist for building server clusters for high availability and load balancing. Few resources exist for detailing how to effectively replicate data in a non-shared storage environment. There are limits to what can be done with today’s technology. These will be discussed in this paper. Our objective is to offer a basic review of what is possible with respect to replicating web content across a set of clustered servers. We also discuss replicating data for other applications (such as databases) and where the future of data replication and synchronization is headed.

This paper contains the following sections:

- Server Clustering
- Shared and Non-Shared Storage Systems
- What is Data Replication & Synchronization?
- Solutions for Replicating Web Data
- Replicating Data for Email, File, and Database Servers
- Future of Data Replication

Server Clustering

Because servers have gotten smaller, more powerful, and less expensive, it is now possible to put them into groups of systems that appear to the customer as a single server system. This group of servers is known as a cluster. The cluster appears to clients as a single system. The cluster enhances total system availability and performance, but can be a management nightmare for the server managers. With a single high-end system such as a mainframe, the server manager need only manage this system. In the server cluster, the manager must install software on multiple hosts, configure and operate each, and insure that data (such as the web site content) is the same on each system. This can be a challenge and is the primary reason that data replication technologies are now becoming important for the market.

There are really three (3) different types of clustering solutions: network clustering, data clustering, and process clustering. Network clustering manages the interface of the network to the cluster. Typically, such products include IP failover, heartbeat, load balancing, and various forms of service and device monitoring. It is this definition of clustering that we are most focused on in this paper. Data Clusters manage access to the shared storage resources of the cluster. Since we are focused on non-shared or
replicated storage in this paper, this form of clustering is not addressed. Finally, process clusters allow multiple nodes to participate in the processing of a single application. Our goal is to show how data replication interacts with network clustering to build a highly available web and application server cluster.

Many of the network clustering solutions on the market are based on a centralized hardware device that front-ends web servers and manager the traffic from the clients to the servers. The problem with this approach is two-fold: First, they are single points of failure and unless they are duplicated they are not redundant. This means they are expensive and complex. The second problem is that since there is nothing on the servers, there is no way to guarantee that data is synchronized across the servers in the cluster. Because of this, we believe that software based clustering solutions will more effectively deal with cluster management and data replication issues. For now, most of the software clustering solutions do not address data synchronization and replication. We believe this will change and that clustering solutions with integrated data replication will begin to make their way to the market. This paper specifically shows how to integrate today’s data replication solutions with server clustering solutions.

Shared and Non-Shared Storage Systems

Most ecommerce and corporate shops use a hierarchical approach to building web and database server applications. In this approach a set of web servers front-ends a database server. This is shown in diagram 1. Even in this approach, although dynamic web content may be created from the shared database server, it is still likely that the basic web content for the web site is retrieved from the local disk rather than from the shared data resource. In the other shared storage alternative shown in diagram 2, there is no database server, the servers act as web and database servers, with both systems having direct access to a shared RAID disk controller that supports dual-porting (say via SCSI interfaces). This alternative has some limitations: usually only one of the servers can mount and access the shared files at once. In both diagram 1 and diagram 2, the shared storage alternatives have a single point of failure and can be costly. In both cases, local web content may be pulled from the local disk rather than the shared storage device or server. The point here is that even though there is a shared storage database server or device, local disks are still used and there can still be an issue of synchronizing the data between all servers in a cluster.

In non-shared storage configurations, there is no shared storage device or there may be a shared database server, but some of the applications on each server in the cluster use local data only. Diagram 3 shows a non-shared storage configuration with the resulting data replication requirement. If the web server cluster is configured in a load balancing or load sharing scenario, users may see different web content if the web files are not replicated and synchronized in real-time. For a set of web servers to be clustered in this manner, the web data must be replicated and synchronized between the servers in real-time. If the cluster is configured with one server as primary and the others as backup, data replication is still required or the backup may be “out of sync” with the primary. If the primary were to fail, the backup host may not have the correct web files. Even in the shared storage examples of Diagram 1 & 2, it is still possible that some degree of replication must exist if applications on each server access data on the local rather than the shared storage medium. Data replication can be important both in shared and non-shared storage configurations.
The types of data files that make up web content may vary greatly; they may include HTML, database, or JPEG/GIF files. The key is that to support a server cluster environment, each of the local disks must have the data “look” the same for all client access. This means the data must be replicated and synchronized on the servers in the cluster.

**What is Data Replication & Synchronization?**

We have seen why data replication and synchronization are needed. Now let’s explain what replication and synchronization really entail, and how it is different for varying applications. For our purposes we define replicated data as data that is fully redundant across all or a subset of servers in a server cluster. This implies that the data is exactly the same on all servers in the cluster. The major issue with respect to replication is the degree to which the data is in fact “synchronized” in time. Synchronization is the process of insuring that the data is fully consistent across all servers in the cluster. In some systems it may not be necessary that the data be synchronized in real-time. In this case a batch job that replicates and synchronizes the files every hour may be fine enough. But in this case, during the hour in which a change is made and the file has yet to be updated on all servers in the cluster, clients may see different data depending upon which server is handling their requests. On the other end of the spectrum, and especially for database transaction systems, it may be necessary that all access to replicated data be the same all of the time. In this case some distributed locking mechanism would need to be employed to guarantee that the data in the replicated database is fully redundant and consistent all of the time. The cost for providing such a service can be high and is usually warranted only when it is absolutely unacceptable to provide anything but 100% data consistency (for instance in the case of financial transactions).

In both of the extremes mentioned above a number of issues must be addresses to deal effectively with some of the problems of data replication and synchronization. The first is disconnected operation. How does the replication software handle servers becoming disconnected (because of problems or normal maintenance)? Does the system recover from disconnected operation? If a subset of the servers becomes disconnected, does the system fail to operate? If the system does operate in a disconnected state, how does it deal with conflicts? If multiple writers make conflicting updates to the data, how does the system deal with these conflicts?

For the purposes of replicating and synchronizing web content, we make the assumption that hourly replication is too slow, but real-time, distributed locking based replication is too costly. We discuss the issues in a near-real time, multiple writer environment and look at some software systems that are available today to solve the web data replication and synchronization problem.
Solutions for Replicating Web Data
This section discusses solutions and examples of data replication software that specifically replicates web content across servers in a cluster. Some of these can support multiple servers in a cluster, some can support less. The three we examine are:

- **Rsync for Linux & Unix**
- **Intermezzo for Linux from Stelias Computing**
- **SureSync Real-Time for Windows NT from Software Pursuits**

Some of these, like rsync, are available now for Unix systems, but aren’t very scalable. Others like Intermezzo have a lot of promise, but really aren’t ready for prime time. SureSync from Software Pursuits is a proven solution for real-time data replication, but is only available for the Windows environments.

**Rsync**
Rsync is an open source utility that provides incremental file transfer capabilities and is ideal for limited mirroring. Rsync is a utility for Unix systems and provides a fast mechanism for synchronizing remote file systems. Rsync is not a real-time mirroring software utility. It must be called by the user either periodically (via a cron job) or manually after updates have been done. The latest version of rsync (2.4.3) is available for a number of platforms, including: Solaris, Linux, BSD, AIX, HPUX, and others. It has even been ported to Windows NT. The binaries for rsync can be downloaded from:

ftp://rsync.samba.org/pub/rsync/binaries/

A number of good examples exist to assist you in configuring your cluster for rsync. Rsync is generally good for mirroring and replicating data in a two (2) server cluster environment. We recommend rsync for those of you who use or are considering using Understudy for load sharing or a primary/backup server configuration in a Linux or Unix (Solaris or FreeBSD) environment.

**Rsync’s features include the following:**
- Ability to update whole directory trees and filesystems
- Can preserve links, file ownership, permissions, devices, and times
- No special privileges to install
- Internal pipelining for multiple files
- Use of rsh, ssh or direct sockets as transport
- Support anonymous rsync for mirroring

Rsync is a very simple utility that does not scale well, but it does have a conflict resolution algorithm based on timestamps. Rsync performs its function by sending just the differences in the files across the network, without requiring that both sets of files are present at one of the ends of the link beforehand. As an example, let’s assume there are two hosts in a server cluster and we have two directories to be replicated. The hosts are MARS and VENUS and the cluster is setup for load sharing using Understudy. The two directories to be replicated are `/home/httpd/html` and `/home/httpd/cgi-bin`. To use rsync we will run a periodic cron job (say every 5 minutes) that would look as follows:

```bash
#!/bin/sh
rsync -avz MARS:/home/httpd/html/* /home/httpd/html/
rsync -avz MARS:/home/httpd/cgi-bin/* /home/httpd/html/cgi-bin
```

This script compresses and then copies all files in the two directories to the MARS node. Note that we would run this script as a cron job on the VENUS host, retaining the attributes of all files (including file permissions, ownership, dates of creation, reference, and updates). Refer to the rsync documentation for details on the parameters and options for rsync. There are a couple of issues with this example that should be noted. First, since the cron job only runs every 5 minutes, that in a load balancing configuration, some users may see out of sync files for up to 5 minutes. The answer may be to run the job more often so that the files are synchronized more often, or to configure the cluster for primary/backup operation rather than load balancing. The other issue is that since the job only runs on VENUS, if updates on VENUS occur and then this server goes down, the updates will not propagate to MARS until VENUS comes back up and the job is run again.
Overall, rsync is a good, do-it-yourself data replication feature that PolyServe recommends for Linux and Unix environments, but can also be used in a Windows environment. Rsync is applicable for a 2 node cluster environment and it does work, but it will take some tuning to get it to work the way you want it to in your environment.

InterMezzo
InterMezzo, from Stelias Computing, Inc., is an open sourced distributed file system with a focus on high availability server clustering. InterMezzo is currently running on Linux (Kernel 2.2 and 2.3). Its primary target is to provide support for flexible replication of directories, with disconnected operation and persistent caching. InterMezzo is based on the Coda File System. You can find the software, documentation, and other information on the InterMezzo web site.

InterMezzo uses a journal based file system to exploit a local filesystem as server storage and as a cache system for other servers to be synchronized with a master filesystem. In a server cluster environment, InterMezzo allows a single master and multiple client server systems. Any write is in real-time replicated amongst the servers in a cluster. InterMezzo supports disconnected operation with reintegration and conflict detection, and server replication. Disconnected operation allows servers to be taken offline for problem or maintenance purposes and once back online, the data is automatically synchronized from the other servers to the now connected server. InterMezzo also has an API that lets users define their own conflict resolution policies.

An excellent white paper, Replicating HTTP Servers, is available from Stelias and overviews the processes and tasks necessary to perform HTTP server replication for a set of clustered web servers. Stelias is also implementing a distributed lock manager to be used in load balancing and clustering Samba, sendmail, imap, as well as http server applications.

The figure above shows what happens when a file folder (say web content) is updated on one of the servers in a cluster (Client 1). The file folder is then updated on the master server in the cluster, and finally the server (Client 2) is updated. Note that InterMezzo works in conjunction with a 3rd party cluster failover mechanism such as PolyServe Understudy.

InterMezzo is in beta at this time, so its use is problematic. But as a technology, it has promise and should be monitored as a solution for web, email, and file application synchronization.

SureSync Real-Time for Windows NT
SureSync Real-Time for Windows NT is a proven, commercially available replication and synchronization technology. SureSync is made by Software Pursuits and they have excellent product information on their web site. There are two versions of SureSync, a batch and a real-time version. Synchronization can occur across multiple servers in a cluster in a two-way or one-way manner. SureSync’s Real-Time product allows a set of logical synchronization rules to be defined. These rules and the GUI that allows the user to configure them allows:

- Flexible selection of folders and files to include/exclude by name, date, or attributes
- Master/Slave or Peer-to-Peer Synchronization to be defined
- Options for mirroring of deletions, new files additions, and target/source updates
- Support for multiple nodes in a one-to-many or many-to-many relationship
- Multiple rules to apply to different sets of files/folders/systems
SureSync allows system and network priorities to be defined for different times of day or day of week. SureSync also has features which play very well with clustering software such as Understudy. This includes the ability to automate recovery and detect failure conditions and perform backup and restoration when the failed system comes back online. Software Pursuits has an excellent clustering white paper, which provides a detailed look at how SureSync works with web servers in a clustered environment.

SureSync has a single point of system control. The SureSync database, containing systems, relations, rules, and synchronization logs can be shared between multiple systems. This provides a single point of control for the system and makes configuring and operating the system easy.

Unlike Rsync and InterMezzo, SureSync is commercially available, supported, and is easy to install, configure, and operate. We highly recommend this solution as a mature method for file replication & synchronization in a Windows NT web server cluster configuration. Software Pursuits also has a 30-day evaluation download capability on their website so that you can get comfortable with their software before purchasing.

**Replicating Data for Email, File, and Database Servers**

Up to this point we have focused on replicating and synchronizing web files across servers in a cluster. What if we would like to replicate and synchronize email, file, or databases across the cluster? As we will see, this is a much more difficult task, one for which there is at present few good solutions.

What if we want to build a cluster of email servers (say just two) so that in case one of the servers goes down, all the data is on the other server and it can pick up where the other server left off? As we shall see, in both a primary/backup and load balancing configuration, this can be difficult to implement. The main reason is that email servers guarantee that mail will always be delivered and that no mail messages will be lost.

Let’s take an example configuration and show how data replication would have to work to maintain this guarantee. Assume two mail servers MailA and MailB. Assume we are in a load balancing scenario in which clients are round robin rotated to each server. Now assume a mail message is sent from the users client machine to MailB and that right after successfully transferring the message, MailB goes down. MailA will not have that mail message and if MailB never comes back up, that message is gone even though the user thinks that it was transferred successfully. If there was a single mail server and it received the message and crashed then it would be clear that the failure scenario was due to a crashed host. In the load balancing case with multiple mail servers, this scenario is unclear. To the user it would appear that the mail server is still up, but that the mail message disappeared. In fact, it would be better if the cluster of mail servers did not report a “successful” delivery until all hosts in the cluster have a copy of the mail message. Providing a data replication solution similar to the web data replication solution will not necessarily meet the needs of email system managers.

File server clustering also presents a bit of a problem for standard data replication technologies discussed in the previous section. Again an example shows the problem. Let’s assume 2 file servers supporting many users accessing files in a real-time, read-write environment. Assume User #1 accesses a file on File Server #1 and attempts to write the file (say it is a Microsoft Word file). Before it has been successfully written, User #2 accesses the same file on File Server #2 in a write mode as well. Then User #1 saves the file changes and User #2 does the same, except that both have now made conflicting updates to the file. In this case User #2’s changes will most likely be accepted since they were the last changes (if timestamps are used for conflict resolution). As we can see, the write-read-write problem with 2 users on a pair of clustered file servers is a concern. If there were a single file server, the second write would be locked out by the filesystem (and the file could only be opened in read-only mode). What is needed is a distributed lock manager. None of the replication technologies we have discussed have this capability. It is possible that a replication scheme could be employed using the technologies we have discussed, but it would only work in a primary/backup configuration, not in a load balanced configuration.

Database server clustering is even more problematic than file server clustering. This is because database files are indexed files and could easily be corrupted and made inconsistent by incorrect file replication technologies. A record level distributed locking mechanism would be needed to guarantee the consistency of the database files. There are some solutions in this space. PeerDirect’s PDRE is one such example of a product that supports a variety of SQL packages in a replication environment (mainly for Windows NT Servers). But it lacks a two-phase commit algorithm so it has some operational problems as well.
The bottom line is that to deliver real data replication and synchronization in a file, email, and database application environment requires more sophistication than in a simple web file replication environment.

Future of Data Replication
There are solutions for web server file replication and synchronization in the Unix, Linux, and Windows NT space. Some of the solutions, like Intellimirror for Windows 2000, purport to be data replication solutions, but in fact aren’t designed for server clusters. Many provide synchronization for hand held devices or support PC to server synchronization. Real server clustering solutions need to deal with disconnected server operation, server reintegration, real-time data replication, and data conflict resolution. Some of the data replication solutions we have discussed are a good start to solving the web server clustering problem. Some like SureSync are fairly mature. But to deal effectively with applications such as samba file servers, email servers, and especially database servers, low-latency, distributed locking mechanisms are required.

PolyServe is committed to finding and/or developing these solutions and integrating them into our product line to provide total server clustering solutions for our customers.