A Survey of Cluster Technologies

Ken Moreau
Solutions Architect, OpenVMS Ambassador, MCSE

Overview

This paper discusses the cluster technologies for the operating systems running on HP server platforms, including HP-UX, Linux, NonStop Kernel, OpenVMS, Tru64 UNIX and Windows 2000. I describe the common functions which all of the cluster technologies perform, show where they are the same and where they are different on each platform, and introduce a method of fairly evaluating the technologies to match them to business requirements. I do not discuss performance, base functionality of the operating systems, or system hardware.

This article draws heavily from the HP ETS 2002 presentation of the same name.

Introduction

Clustering technologies are highly inter-related, with almost everything affecting everything else. But I have broken this subject into five areas:

- Single/multi-system views, which defines how you manage and work with a system, whether as individual systems or as a single combined entity.
- Cluster file systems, which defines how you work with storage across the cluster. Cluster file systems are just coming into their own in the UNIX world, and I will talk about how they work in detail.
- Configurations, which defines how you assemble a cluster, both physically and logically.
- Application support, which discusses how applications which are running on your single standalone system today, can take advantage of a clustered environment. Do they need to change, and if so how? What benefits are there in a clustered environment?
- Resilience, which talks about when bad things happen to good computer rooms. This covers things like host-based RAID, wide area "stretch" clusters, extended clusters, and disaster tolerant scenarios.

I cover the capabilities of Linux LifeKeeper, NonStop Kernel G06.13, Serviceguard 11i both for HP-UX and Linux, TruCluster V5.1b, OpenVMS Cluster Software V7.3-1, and Windows 2000 DataCenter system clusters.

For Linux, I focus on the High Availability side, not the HPTC (i.e., Beowulf) technologies.

Single-System and Multi-System-View Clusters

In order to evaluate the cluster technologies fairly, we need to define four terms: scalability, reliability, availability and manageability.

- Availability defines whether the application stays up, even when components of the cluster go down. If I have two systems in a cluster and one goes down but the other picks up the workload, that application is available even though half of my cluster is down. Part of availability is failover time, because if it takes 30 seconds for the application to fail over to the

other system, the users on the first system think that the application is down for those 30 seconds.

- Reliability defines how well the system performs during a failure of some of the components. If I get sub-second query response and if my batch job finishes in 8 hours with all of the systems in the cluster working properly, do I still get that level of performance if one or more of my systems in the cluster is down? If I have two systems in a cluster and each system has 500 active users with acceptable performance, will the performance still be acceptable if one of the systems fails and there are now 1,000 users on a single system? Keep in mind that your users neither know nor care how many systems there are in your cluster, they simply care whether they can rely on the environment to get their work done.

  Notice that reliability and availability are orthogonal concepts, where you can have one but not the other. How many times have you logged into a system (i.e., it was available), but it was so slow as to be useless (i.e., it was not reliable)?

- Scalability defines the percentage of useful performance you get from a group of systems. For example, if I add a second system to a cluster, do I double my performance, or do I get a few percentage points less than that? If I add a third, do I triple the performance of one, or not?

- Manageability tells us how much additional work it is to manage those additional systems in the cluster. If I add a second system to my cluster, have I doubled my workload because now I have to do everything twice? Or have I added only a very small amount of work, because I can manage the cluster as a single entity?

Multi-system-view clusters are generally comprised of two systems, where each system is dedicated to a specific set of tasks. Storage is physically cabled to both systems, but each file system can only be mounted on one of the systems. This means that the applications cannot simultaneously access data from both systems at the same time. It also means that the operating system files cannot be shared between the two systems, so there needs to be a fully independent boot device (called a "system root" or "system disk") with a full set of operating system and cluster software files for each system.

Multi-System-View Clusters in an Active-Passive Mode

Multi-system-view clusters in an active-passive mode are the easiest for vendors to implement. They simply have a spare system on standby in case the original system fails in some way. The spare system is idle most of the time, except in the event of a failure of the active system. This is why it is called active-passive, because one of the systems is active and the other is not during normal operations. This is classically called N+1 clustering, where N=1 for a two system cluster. For clusters with larger numbers of systems, you would have a spare server that could take over for any number of active systems.

Failover can be manual or automatic. Because both systems are cabled to the same storage array, the spare system will be monitoring the primary system and can start the services on the secondary system if it detects a failure of the primary system. The “heartbeat” function can be over the network or by some private interface.

In a multi-system-view cluster in an active-passive mode, the availability, reliability, scalability, and manageability characteristics can be described as follows:

- Availability is increased because you now have two systems available to do the work. The odds of both systems being broken at the same time are fairly low but still present.

- Reliability can be perfect in this environment, because if the two systems are identical in terms of hardware, the application will have the same performance no matter which system it is running on.

- Scalability is poor (non-existent?) in an active-passive cluster. Because the applications cannot access a single set of data from both systems, there is no scalability in a multi-system-view cluster. You have two systems' worth of hardware doing one system's worth of work.
- Manageability is poor, because it takes approximately twice as much work to manage as that of a single system. Because there are two system roots, any patches or other updates need to be installed twice, backups need to be done twice, etc. Further, you have to test the failover and failback, which adds to the system management workload

Notice that the second system is idle most of the time, and you are getting no business benefit from it. So the other alternative is to have both systems working.

Multi-System-View Clusters in an Active-Active Mode

The physical environment of a multi-system-view cluster in an active-active mode is identical to that of the active-passive mode. There are generally two systems, physically cabled to a common set of storage, but only able to mount each file system on one of the systems. The difference in this case is there are multiple systems that are performing useful work as well as monitoring each other's health. However, they are not running the same application on the same data, because they are not sharing anything between the systems.

For example, a database environment could segment their customers into two groups, such as all people with last names from A-M and from N-Z. Then each group would be set up on a separate partition on the shared storage, and each system would handle one of the groups. This is known as a "federated" database. Or one of the systems could be running the entire database and the other system could be running the applications that access that database.

In the event of a failure, one system would handle both groups.

This is called an N+M cluster, because any of the systems can take over for any of the other systems. One way to define N and M is to think about how many tires you have on your automobile. Most people automatically say four, but then realize that they really have five tires, operating in an N+1 environment, because four tires are required for minimum operation of the vehicle. A variation is to use the "donut" tires, which offer limited performance but enough to get by. This can be thought of as having 4½ tires on the vehicle. The key is to define what level of performance and functionality you require, and then define N and M properly for that environment.

Failover can be manual or automatic. The "heartbeat" function can be over the network or by some private interface.

In a multi-system-view cluster in an active-active mode, the availability, reliability, scalability, and manageability characteristics can be described as follows:

- Availability is increased because you now have two systems available to do the work. Just as in the active-passive environment, the odds of both systems being broken at the same time are fairly low but still present.
- Reliability is not guaranteed in this situation. If each system is running at 60% of capacity, then a failure will force the surviving system to work at 120% of capacity, and we all know what happens when you exceed about 80% of capacity.
- Scalability is poor in this situation because each workload must still fit into one system. There is no way to spread a single application across multiple systems.
- Manageability is slightly worse than the active-passive scenario, because you still have two independent systems, as well as the overhead of the failover scripts and heartbeat.

Failover of a Multi-System-View Cluster (Active-Active or Active-Passive)

One of the factors affecting availability is the amount of time it takes to accomplish the failover of a multi-system-view cluster, whether active-active or active-passive. The surviving system must:

- Notice that the other system is no longer available, which is when the "heartbeat" function on the surviving system does not get an answer back from the failed system.
- Mount the disks that were on the failing system. Remember that the file systems are only mounted on one system at a time: this is an unbreakable rule, and a definition of multi-system-view clusters. So the surviving system must mount the disks that were mounted on the other system. If you have a large number of disks, or large RAIDsets, this could take a while.
- Start the applications that were active on the failing system.
- Initiate the recovery sequence for that software. For databases, this might include processing the re-do logs in order to process any in-flight transactions which the failing system was performing at the time of the failure.

In large environments, it is not unusual to have this take 30-60 minutes. And during this recovery time, the applications that were running on the failed system are unavailable, and the applications that were running on the surviving system are suffering from reliability problems, because the single system is now doing much more work.

Single-System-View Clusters

In contrast, single-system-view clusters offer a unified view of the entire cluster. All systems are physically cabled to all storage and can directly mount all storage on all systems. This means that all systems can run all applications, see the same data on the same partitions, and cooperate at a very low level. Further, it means that the operating system files can be shared in a single "shared root" or "shared system disk," reducing the amount of storage and the amount of management time needed for system maintenance. There are no spare systems. All systems can run all applications at all times. And in a single-system-view cluster, there can be many systems. In a single-system-view cluster, the availability, reliability, scalability, and manageability characteristics can be described as follows:

- Availability is increased because you now have multiple systems to do the work. The odds of all systems being broken at the same time is now much lower, because you can have potentially many systems in the cluster.
- Reliability is much better, because with many systems in the cluster, a failure of a single system will allow that workload to be spread across many systems, increasing their load only slightly. For example, if each system is running at 60% capacity and one server out of four fails, you would take 1/3 of the load and place it on each of the other systems, increasing their performance to 80% of capacity, which will not affect reliability significantly.
- Scalability is excellent because you can spread the workload across multiple systems. So if you have an application which is simply too big for a single computer system (even one with 64 or 128 CPUs and hundreds of gigabytes of memory and dozens of I/O cards), you can have it running simultaneously across many computer systems, each with a large amount of resources, all directly accessing the same data.
- Manageability is much easier than the equivalent job of managing this number of separate systems, because the entire cluster is managed as a single entity. So there is no increase in management workload even when you have lots of smaller systems.

Notice how this gives some advantages in failover times over multi-system-view clusters by not having to do quite so much work during a failover:

- The surviving system must detect the failure of the other system. This is common between the two schemes.
- The surviving systems do not have to mount the disks from the failed system: they are already mounted.
- The surviving systems do not have to start the applications: they are already started.
- The execution of the recovery script, is common between the two schemes, and can begin almost instantly in the single-system-view cluster case. The application recovery time will be similar on both systems, but if you have a larger number of smaller systems, you can achieve parallelism even in recovery, which means your recovery might be faster in this case as well.

How Do the Clusters on HP Systems Fit into These Schemes?

So now that we understand the terms, how do the clusters on HP systems fit into these schemes?

	Multi-system view	Single-system view	Shared root
LifeKeeper Linux, Windows	Yes	No	No
Serviceguard	Yes	No	No
NonStop Kernel	Yes	Yes	Each node (16 CPUs)
TruCluster	No	Yes	Yes
OpenVMS Cluster Software	Yes	Yes	Yes
Windows 2000 DataCenter	Yes	No	No

Linux Clustering

Linux clustering is split between massive system compute farms (Beowulf and others) and a multi-system-view, failover clustering scheme. I am not talking about the High Performance Technical Computing market here, which breaks down a massive problem into many (hundreds or thousands) of tiny problems and hands them off to many (hundreds or thousands) of small compute engines. The point is that this is not a high availability environment, because if any of those compute engines fails, that piece of the job has to be restarted from scratch.

The high availability efforts going on with SuSE and others are focused on multi-system-view clusters consisting of two systems so that applications can fail over from one system to the other. I will talk later about some cluster file system projects such as Lustre, GFS and PolyServe, but these do not offer shared root, so systems in Linux clusters require individual system disks.

I will not be discussing the work being done by HP as part of the Single System Image Linux project, because it is not yet a product. But when this becomes a product, it will have significant capabilities that match or exceed every other clustering product.

Serviceguard

Serviceguard is a multi-system-view failover cluster. Each system in a Serviceguard cluster requires its own system disk. There are excellent system management capabilities via the Service Control Manager and the Event Management Service, including the ability to register software in the System Configuration Repository, get system snapshots, compare different systems in the cluster, etc. It is also well integrated with HP/OpenView.

Himalaya NonStop Kernel

The Himalaya NonStop Kernel can be configured either as a multi-system-view cluster or, more commonly, as a single-system-view cluster. It offers the best scalability in the industry, in fact, true linear scalability because of the superb cluster interconnect, both hardware and software. Each cabinet of up to 16 processors can share a system disk and is considered one system.

TruCluster V5.1b

TruCluster V5.1b represents a major advance in UNIX clustering technology. It can only be configured as a single-system-view, with all of the focus on clustering being to manage a single system or a large cluster in exactly the same way, with the same tools, and roughly the same amount of effort. It offers a fully shared root, with a single copy of almost all system files.

OpenVMS Cluster Software

OpenVMS Cluster Software has always been the gold standard of clustering. It also can be configured as either multi-system view or single-system view, although the most common is single-system view. It supports a single or multiple system disks.

Windows 2000 DataCenter

Windows 2000 DataCenter is a multi-system-view failover scheme. Applications are written to fail over from one system to another. Each system in a Windows 2000 DataCenter cluster requires its own system disk. This is not to say that there aren't tools like the Cluster Administrator, which can ease some of this burden, but they are still separate system disks that must be maintained.

Cluster File Systems

Cluster file systems are how systems communicate with the storage subsystem in the cluster. There are really two technologies here: how a group of systems communicates with volumes that are physically connected to all of the systems, and how a group of systems communicates with volumes that are only physically connected to one of the systems.

Network I/O allows all of the systems in a cluster to access data, but in a very inefficient way that does not scale well. Let's say that volume A is a disk or tape drive which is physically cabled to a private IDE or SCSI adapter on system A. It cannot be physically accessed by any other system in the cluster, so if any other system in the cluster wants to access files on it, it must do network I/O, usually by some variation of NFS.

Specifically, if system B wants to talk to the device that is mounted on system A, the network client on system B communicates to the network server on system A, in the following way:

- An I/O is initiated across the cluster interconnect from system B to system A.
- System A receives the request, and initiates the I/O request to the volume.

- System A gets the data back from the volume, and then initiates an I/O back to system B.

Notice that there are three I/Os for each disk access. For NFS, there is also significant locking overhead with many NFS clients. This leads to poor I/O performance with an active/active system.

So why does every system offer network I/O? To deal with single-user devices that cannot be shared, such as tapes, CD-ROM, DVD or diskettes, and to allow access to devices that are on private communications paths, such as disks on private IDE or SCSI busses.

In contrast, direct access I/O (also known as concurrent I/O) means that each system is able to independently access any and all devices, without bothering any other node in the cluster. Notice that this is different from direct I/O, which simply bypasses the file systems cache. Most database systems do direct I/O both in a clustered and non-clustered environment, because they are caching the data anyway, and don't need to use the file systems cache.

Implementing direct access I/O allows a cluster file system to eliminate two out of three I/Os involved in the disk access in network I/O, because each system talks directly over the storage interconnect to the volumes. It also provides full file system transparency and cache coherency across the cluster.

Now, you may object that we could overwhelm a single disk with too many requests. Absolutely true, but this is no different from the same problem with other file systems, whether they are clustered or not. Single disks, and single database rows, are inevitably going to become bottlenecks. You design and tune around them on clusters in exactly the same way you design and tune around them on any other single member operating system, using the knowledge and tools you use now.

The I/O attributes of HP cluster offerings are summarized in the following table.

	Network I/O	Direct Access I/O	Distributed Lock Manager
LifeKeeper Linux, Windows	NFS	Oracle raw devices, GFS	Supplied by Oracle
Serviceguard	Yes	Oracle raw devices	OPS Edition
NonStop Kernel	Data Access Manager	Effectively Yes	Not applicable
TruCluster	Device Request Dispatcher	Cluster File System	Yes
OpenVMS Cluster Software	Mass Storage Control Protocol	Files-11 on ODS-2 or -5	Yes
Windows 2000 DataCenter	NTFS	Supplied by Oracle	Supplied by Oracle

Pretty much every system in the world can do client/server I/O, here called network I/O. And this makes sense, because every system in the world has the problem of sharing devices that are not on shared busses, so you need to offer this functionality.

FailSafe and Serviceguard do it with NFS or Veritas, NonStop Kernel does it with the Data Access Manager (DAM), TruCluster does it both with the Device Request Dispatcher (DRD) and the Cluster File System, OpenVMS Cluster Software does it with the Mass Storage Control Protocol (MSCP), and Windows 2000 does it with NTFS and Storage Groups.

The more interesting case is direct access I/O. One point I need to make here is that Oracle offers direct access I/O on raw devices on almost every system they support. Raw devices are not as functional as file systems, but they do exist and they do (at least with Oracle) offer direct access I/O on every major computing platform in the industry.

One of the Linux projects being done by HP and Cluster File Systems Inc for the US Department of Energy will enhance the Lustre File System originally developed at Carnegie Mellon University. It is focused on high-performance technical computing environments. Oracle has developed a cluster file system for the database files for Linux, as part of Oracle 9i Real Application Clusters 9.2. It is a layer on top of raw devices.

NonStop Kernel is interesting, because, strictly speaking, all of the I/O is network I/O, and yet because of the efficiencies and reliability of NSK, and the ability of NSK to transparently pass ownership of the volume between CPUs, it shows all of the best features of direct access I/O without the poor performance and high overhead of all other network I/O schemes. So, effectively, NSK offers direct access I/O, even though it is done using network I/O. The NonStop Kernel (effectively NonStop SQL) utilizes a "shared-nothing" data access methodology. Each processor owns a subset of disk drives whose access is controlled by processes called the Data Access Managers (DAM). The DAM controls and coordinates all access to the disk so a DLM is not needed.

Serviceguard and Windows 2000 DataCenter do not offer a direct access I/O methodology of their own, but rely on Oracle raw devices. Oracle has developed a cluster file system for Windows 2000 DataCenter for the database files, as part of Oracle 9i Real Application Clusters 9.2. It is a layer on top of raw devices.

TruCluster offers a cluster file system (CFS) which allows transparent access to any file system from any system in the cluster. However, all write operations, as well as all read operations on files smaller than 64KBytes, are done by the CFS server system upon request by the CFS client systems. In effect, all write operations and all read operations on small files are performed using network I/O. The only exception to this is applications which open the file with O_DIRECTIO.

OpenVMS Cluster Software extends the semantics of the Files-11 file system transparently into the cluster world. A file which is opened for shared access by two processes on a single system, and the same file which is opened for shared access by two processes on two different systems in a cluster, will act identically. In effect, all file operations are automatically cluster-aware.

Every operating system has a lock manager for files in a non-clustered environment. A distributed lock manager simply takes this concept and applies it between and among systems. Oracle, because they need to run in the same way across many operating environments, developed their own distributed lock manager, which is available on Linux and Windows systems.

Serviceguard includes a distributed lock manager as part of the Serviceguard Extension for RAC.

NSK does not even have the concept of a distributed lock manager. All resources (files, disk volumes, etc) are local to a specific CPU, and all communication to any of those resources is done via the standard messaging between CPUs, so any locking required is done locally to that CPU. But again, because of the efficiencies of the implementation, this scales superbly well.

TruCluster has the standard set of UNIX APIs for local locking, and a separate set of APIs that were implemented to allow applications to perform distributed locking. Applications need to be modified to use the distributed locking APIs in order to become cluster-aware.

OpenVMS Cluster Software uses the same locking APIs for all locks, and makes no distinction between local locks and remote locks. In effect, all applications are automatically cluster-aware.

Quorum

Before discussing cluster configurations, it is important to understand the concept of quorum. Quorum devices (which can be disks or systems) are a way to break the tie when two systems are equally capable of forming a cluster and mounting all of the disks, and will prevent cluster partitioning.

When a cluster is first configured, you assign each system a certain number of votes, generally 1. Each cluster environment defines a value for the number of "expected votes" that there should be for optimal performance. This is almost always the number of systems in the cluster. From there, we can calculate the "required quorum" value, which is the number of votes that are required to form a cluster. If the "actual quorum" value is below this number, the software will refuse to form a cluster, and will generally refuse to run at all.

For example, assume there are two members of the cluster, system A and system B, each with one vote, so the required quorum of this cluster is 2.

Now in a running cluster, expected votes is the sum of all of the members with which the connection manager can communicate. Since the cluster interconnect is working, there are 2 systems available and no quorum disk, so this value is 2. So actual quorum is greater than or equal to required quorum, and we have a valid cluster.

But what happens if the cluster interconnect fails? The cluster is broken, and a cluster transition occurs.

The connection manager of system A cannot communicate with system B, so actual votes becomes 1 for each of the systems. Applying the equation, actual quorum becomes 1, which is less than the number of required quorum that is needed to form a cluster, so both systems stop and refuse to continue processing.

But what would happen if one or both of the systems continue on its own? Because there is no communication between the systems, they would both try to form a single system cluster, as follows:

- System A decides to form a cluster, and mounts all of the disks.
- But system B also decides to form a cluster, and also mounts all of the disks. This is called cluster partitioning.
- The disks are mounted on two systems that cannot communicate with each other. This usually leads to instant disk corruption, as they both try to create, delete, extend and write to files without doing things like locking or cache coherency.

So how do we avoid this? A quorum device.

Same configuration as before, but here we have added a quorum disk, which is physically cabled to both systems. Each of the systems has one vote, and the quorum disk has one vote. The connection manager of system A can communicate with system B and with the quorum disk, so expected votes is 3. This means that the quorum is 2. But the cluster interconnect fails again. So what happens this time?

- Both systems attempt to form a cluster, but system A wins the race and accesses the quorum disk first. Because it cannot connect to system B, and the quorum disk watcher on system A

observes that at this moment there is no remote I/O activity on the quorum disk, system A becomes the founding member of the cluster, and writes into the quorum disk things like the system id of the founding member of the cluster, and the time that the cluster was newly formed. System A then computes the votes of all of the cluster members (itself and the quorum disk for a total of 2) and observes it has sufficient votes to form a cluster. It does so, and then mounts all of the disks on the shared bus.

- System B comes in second and accesses the quorum disk. Because it cannot connect to system A, it thinks it is the founding member of the cluster, so it checks this fact with the quorum disk, and discovers that system A is in fact the founding member of the cluster. But system B cannot communicate with system A, and as such, it cannot access the quorum disk. So system B then computes the votes of all of the cluster members (itself only, so only one vote) and observes it does not have sufficient votes to form a cluster. Depending on other settings, it may or may not continue booting, but it does not attempt to form or join the cluster. There is no partitioning of the cluster.

In this way only one of the systems will mount all of the disks in the cluster. If there are other systems in the cluster, the value of required quorum and expected quorum would be higher, but the same algorithms will allow those systems that can communicate with the founding member of the cluster to join the cluster, and those systems which cannot communicate with the founding member of the cluster to be excluded from the cluster.

Cluster Configurations

The following table summarizes important configuration characteristics of HP cluster technologies.

	Max Systems In Cluster	Cluster Interconnect	Quorum Device
LifeKeeper Linux, Windows	16	Network, Serial	Yes (Optional)
Serviceguard	16	Network, HyperFabric	Yes = 2, optional >2
NonStop Kernel	255	SystemNet, TorusNet	No
TruCluster	8 generally, 32 w/Alpha SC	100Enet, QSW, Memory Channel	Yes (Optional)
OpenVMS Cluster Software	96	CI, Network, MC, Shared Mem	Yes (Optional)
Windows 2000 DataCenter	4	Network	Yes

Linux is focusing on multi-system-view failover clusters, so FailSafe supports up to 16 systems in a cluster. These systems are connected by either the network or by serial cables. Any of the systems can take over for any of the other systems. Quorum disks are supported but not required.

Serviceguard can have up to 16 systems, using standard networking or HyperFabric as a cluster interconnect. The one special requirement here is that all cluster members must be present to initially form the cluster (100% quorum requirement), and that >50% of the cluster must be present in order to continue operation. The quorum disk or quorum system is used as a tie breaker when there is an even number of production systems and a 50/50 split is possible. For two nodes, a quorum device is required, either a disk or system. Quorum devices are optional for any other size cluster. Cluster quorum disks are supported for clusters of 2-4 nodes, and cluster quorum systems are supported for clusters of 2-16 systems.

NonStop Kernel can have up to 255 systems in the cluster, but given the way the systems interact, it is more reasonable to say that NonStop Kernel can have 255 systems * 16 processors in each system = 4,080 processors in a cluster. SystemNet is used as a communications path for each system, with TorusNet providing the cluster interconnect to the legacy K-series, and SystemNet providing a more modern cluster interconnect for the S-series, including remote datacenters. NSK does not use the quorum scheme, because it is a shared nothing environment where the quorum scheme does not make any sense.

TruCluster can have up to eight systems of any size in a cluster. For the HPTC market, the Alpha System SuperComputer system farm can have up to 32 systems. This configuration uses the Quadrix Switch (QSW) as an extremely high-speed interconnect.

OpenVMS Cluster Software supports up to 96 systems in a cluster, spread over multiple datacenters up to 500 miles apart. Each of these can also be any combination of VAX and Alpha systems, or, in 2004, any combination of Itanium and Alpha systems, for mixed architecture clusters.

TruCluster and OpenVMS Cluster Software recommend the use of quorum disks for 2 node clusters but make it optional for clusters with larger numbers of nodes.

Windows 2000 DataCenter can have up to four systems in a cluster, but keep in mind that Windows 2000 DataCenter is a services sale, and only Microsoft qualified partners like HP can configure and deliver these clusters. The only cluster interconnect available is standard LAN networking.

Application Support

The following table summarizes application support provided by the HP cluster technologies.

	Single-instance (failover mode)	Multi-instance (cluster-wide)	Recovery Methods
LifeKeeper Linux, Windows	Yes	No	Scripts
Serviceguard	Yes	No	Packages and Scripts
NonStop Kernel	Yes (takeover)	Effectively Yes	Paired Processing
TruCluster	Yes	Yes	Cluster Application Availability
OpenVMS Cluster Software	Yes	Yes	Batch /RESTART
Windows 2000 DataCenter	Yes	No	Registration, cluster API

Single-Instance and Multi-Instance Applications

When talking about applications and clusters, there are really two ways to think about it: single-instance applications and multi-instance applications. Notice that these terms are the opposite of multi-system-view and single-system-view cluster terms that we started with, but those are the terms we are stuck with.

Multi-system-view clusters allow single-instance applications, which offer failover of applications for high availability, but don't allow the same application to work on the same data on the different systems in the cluster.

Single-system-view clusters allow multi-instance applications, which offer failover for high availability, but also offer cooperative processing, where applications can interact with the same data and each other on different systems in the cluster.

A good way to determine if an application is single-instance or multi-instance is to run the application in several processes on a single system. If the applications do not interact in any way, and therefore run properly whether there is only one process running the application or every process on a single system is running the application, then the application is single-instance.

An example of a single-instance application is telnet, where multiple systems in a cluster can offer telnet services, but the different telnet sessions themselves do not interact with the same data or each other in any way. If a given system fails, the user simply logs in to the next system in the cluster and re-starts their session. This is simple failover. And this is the way that many systems including Linux and Windows 2000 DataCenter, and all of our competitors, set up NFS services, as a single instance application in failover mode.

If, on the other hand, the applications running in the multiple processes interact properly with each other, such as by sharing cache or locking data structures to allow proper coordination between the application instances, then the application is multi-instance.

An example of a multi-instance application is a cluster file system that allows multiple systems to all offer the same set of disks as services to the rest of the environment. Having one set of disks being offered by multiple systems requires a cluster file system with a single-system-view cluster, which can be offered either in the operating system software itself (TruCluster and OpenVMS Cluster Software) or by third party software (Oracle 9i Real Application Clusters). So, even though Linux FailSafe, Serviceguard and Windows 2000 get a "No" for multi-instance applications, Oracle is the exception to this. As I mentioned before, NonStop Kernel does this in a different way, but offers effectively the same functionality.

Recovery Methods

Recovery methods implement how the cluster will recover the applications that were running on a system which has been removed from the cluster, either deliberately or by a system failure. For multi-system-view clusters like Linux, this is done by scripts which are invoked when the heartbeat messages between the cluster members detects the failure of one of the systems. Two examples for Linux are 'mon' and 'monit.'

For fault tolerant systems like NonStop Kernel, recovery is done by paired processes, where a backup process is in lockstep with a primary process, ready to take over in the event of any failure.

Serviceguard has extensive tools to group applications and the resources needed to run them, into "packages" which are then managed as single units. The system administrator can define procedures to recover and re-start the application packages onto another system in a cluster in the event of server failure. This is one of the ways Serviceguard leads the industry in UNIX clustering technology.

Both TruCluster and OpenVMS Cluster multi-instance applications have built-in methods that enable recovery from failing systems. Both TruCluster and OpenVMS Cluster Software can monitor some applications and recover them automatically. TruCluster does this via the Cluster Application Availability facility, and OpenVMS Cluster Software does this by the /RESTART switch on the batch SUBMIT command.

There are three main recovery methods with Windows 2000 Datacenter:

- Generic application/generic service. This doesn't require development of any kind, simply a one-time registration of the application for protection by Windows 2000 Datacenter. There is also a wizard to guide administrators through this process step by step.
- Custom resource type. The application itself is unchanged, but the application vendor (or other party) develops a custom resource DLL that interfaces an application with Windows 2000 Datacenter to do application-specific monitoring and failover. Again, this doesn't require any development at integration time, simply registration of the application using the custom resource DLL.
- Cluster API. Here the application is modified to explicitly comprehend that it's running in a clustered environment and can perform cluster-related operations, e.g., failover, query nodes, etc.

Cluster Resilience

The following table summarizes cluster resilience characteristics of HP cluster technologies.

	Dynamic Partitions	Data High Availability	Disaster Tolerance
LifeKeeper Linux, Windows	No	Distributed Replicated Block Device (DRBD)	Extended Mirroring
Serviceguard	vPars	Multi-Path I/O (a) MirrorDisk/UX	Extended Clusters
NonStop Kernel	No	Multi-Path I/O (p), RAID-1, Process Pairs	Remote Database Facility
TruCluster	No	Multi-Path I/O (a) LSM RAID-1	StorageWorks Continuous Access
OpenVMS Cluster Software	Galaxy	Multi-Path I/O (p) HBVS RAID-1	DTCS, StorageWorks Continuous Access
Windows 2000 DataCenter	No	SecurePath (p) NTFS RAID-1	StorageWorks Continuous Access

Dynamic Partitions

One of the major selling points of clusters is high availability, even when things go wrong, such as storage subsystem failures or peak workloads beyond expectations, and even when things go very wrong, such as physical disasters. So how do clusters on HP systems cope with these things?

Dynamic partitions protect against peaks and valleys in your workload. Traditionally you built a system with the CPUs and memory for the worst-case workload, accepting the fact that this extra hardware would be unused most of the time. And with hard partitioning becoming more popular because of system consolidation, each hard partition would require enough CPUs and memory for the worst-case workload. But dynamic partitioning lets you share this extra hardware between partitions of a larger system, so that, for example, you can allocate the majority of your CPUs to the on-line processing partition during the day, and move them to the batch partition at night. Only HP-UX with vPars and OpenVMS with Galaxy offers this functionality. Both offer the ability to dynamically move CPUs between the partitions either via a GUI or the command line. Galaxy offers the ability to share physical memory that is available to two or more Galaxy partitions. HP-UX offers the capability for the Work Load Management (WLM) tools to move the CPUs in response to goal-based operational requirements.

Notice that all of the above software runs in the base operating systems, not just in the clustering products. Both vPars and Galaxy partitions can be clustered just as any other instances of the respective operating systems can be clustered.

Data High Availability

Data high availability assumes that all the proper things are being done at the storage sub-system level, such as redundant host adapters, redundant paths from the host adapters to the storage controllers (through redundant switches if you are using FibreChannel), redundant storage controllers configured for automatic failover, and the appropriate RAID levels on the disks themselves. And some of this redundancy requires cooperation from the host operating system, specifically in the area of multi-path I/O.

Multi-path I/O allows the system to have multiple physical paths from the host to a specific volume, such as multiple host adapters connected to redundant storage controllers. This is extremely common with FibreChannel, but is also achievable with SCSI.

Support for multi-path I/O can be either active or passive. Active multi-path I/O means that both paths are active at the same time, and the operating system can load balance the I/O requests between the multiple physical pathsby choosing the host adapter that is least loaded for any given I/O. In the event of a path failure (caused by the failure of the host adapter or a switch or a storage controller), the operating system would simply re-issue the I/O request to another path. This action would be transparent to the application.

Passive multi-path I/O means that only one path is active at one time, but the other path is ready to take over in the event of the first path failing. This is accomplished in the same way as the active multi-path I/O, by having the system re-issue the I/O request. The above chart shows whether the operating system supports active (a) or passive (p) multi-path I/O.

But all of these technologies are not adequate if you have multiple simultaneous disk failures, such as by physical destruction of a storage cabinet.

The first level of defense against this is host-based RAID, which performs mirroring or shadowing across multiple storage cabinets. Linux does it with Distributed Replicated Block Device (DRBD), where one of the systems writes to a local disk and then sends an update to the other system over the network so it can write a copy of that data to its local disk.

HP-UX uses MirrorDisk/UX to maintain up to three copies of the data. The software needed to enable active multi-path I/O varies depending on the storage system being used by HP-UX. PowerPath is an EMC product, which is compiled into the HP-UX kernel to give active multi-path I/O to EMC storage arrays, where the HP Logical Volume Manager (LVM) gives active multi-path I/O to the XP and EVA storage sub-systems.

NonStop Kernel provides data high availability using a combination of RAID-1 (mirrored disks), active multi-path I/O with multiple SystemNet Fabrics, multiple controller paths, etc, and process pair technology for the fault tolerant Data Access Managers (DAM).

Tru64 UNIX supports RAID-1 and RAID-5 with the Logical Storage Manager, which can protect any disk including the system root. LSM also supports active multi-path I/O.

OpenVMS supports RAID-1 with Host-Based Volume Shadowing, which can maintain up to three copies of any disk including the system disk. OpenVMS supports passive multi-path I/O, with operator controlled load balancing.

Windows 2000 DataCenter does it with NTFS mirroring. Many storage vendors, including HP, offer software with their storage offerings, which layers on top of Windows 2000 to allow passive multi-path I/O. On Windows, the software is called SecurePath.

Disaster Tolerance

Disaster tolerance is what you need to protect computer operations and data from physical disasters. In my area of Florida, we worry about hurricanes. In other areas of the world, we worry about tornadoes or blizzards. And everybody worries about earthquakes, power failures and fires. The only way to protect from these things is to make sure that your data is stored somewhere far away, and to do this in as close to real-time as possible. There are multiple kinds of data replication, but the two major ones are physical replication and logical replication.

Physical replication can be done by either the system or the storage sub-system. The systems use the same software that is used for data high availability detailed above, except that the second (or in some cases, the third) copy of the data is in another physical location. Serviceguard uses MirrorDisk/UX, NonStop Kernel uses the Remote DataCenter Facility (RDF), TruCluster uses LSM and OpenVMS Cluster Software uses host-based Volume Shadowing. The rules for accessibility of the remote volumes by the remote systems are the same as if the remote volumes and the systems were in the same room as the source systems and volumes. Only TruCluster and OpenVMS Cluster Software can share volumes between systems, whether local or remote. Having the operating system accessing volumes across a wide area FibreChannel system is called an “extended cluster”.

We have the same situation here as we had for RAID, in that the storage sub-system offers significant capabilities for physical replication, regardless of the host operating system or clustering software capabilities. StorageWorks Continuous Access for the EVA and the XP storage arrays support the clusters of HP environments and most competitive UNIX systems.

Both the system-based data high availability software and the storage-based Continuous Access offers active/active bi-directional data replication. Exactly what does that mean?

Assume you have a production system in Los Angeles, which is working fine. You want to safeguard your data, so you decide to build a disaster recovery site in San Bernardino, about 100km (60 miles) away from Los Angeles.

First, you put in a group of FibreChannel switches, and connect them using the FibreChannel to ATM adapters to the FibreChannel switches in your Los Angeles data center. Then you put a duplicate set of storage in San Bernardino, and begin replicating the production system’s data from Los Angeles to San Bernardino. This is known as active/passive replication, because San Bernardino is simply a data sink: no processing is going on there, mostly because there are no systems on that side.

This works well, but as times goes on you need more than this: you need processing to continue in San Bernardino even if your Los Angeles data center is wiped out. So you put some systems in San

Bernardino, and physically cable them to the storage. But the target volume of the Continuous Access copy is not available for mounting by the systems, no matter what operating system they are running, so the San Bernardino site is idle, simply waiting for a signal to take over the operations from Los Angeles. This signal can be automated or manual, but in either case, the storage sub-system would break the Continuous Access link, the systems would mount the volumes, and would then initiate any recovery mechanisms that have been defined for the application.

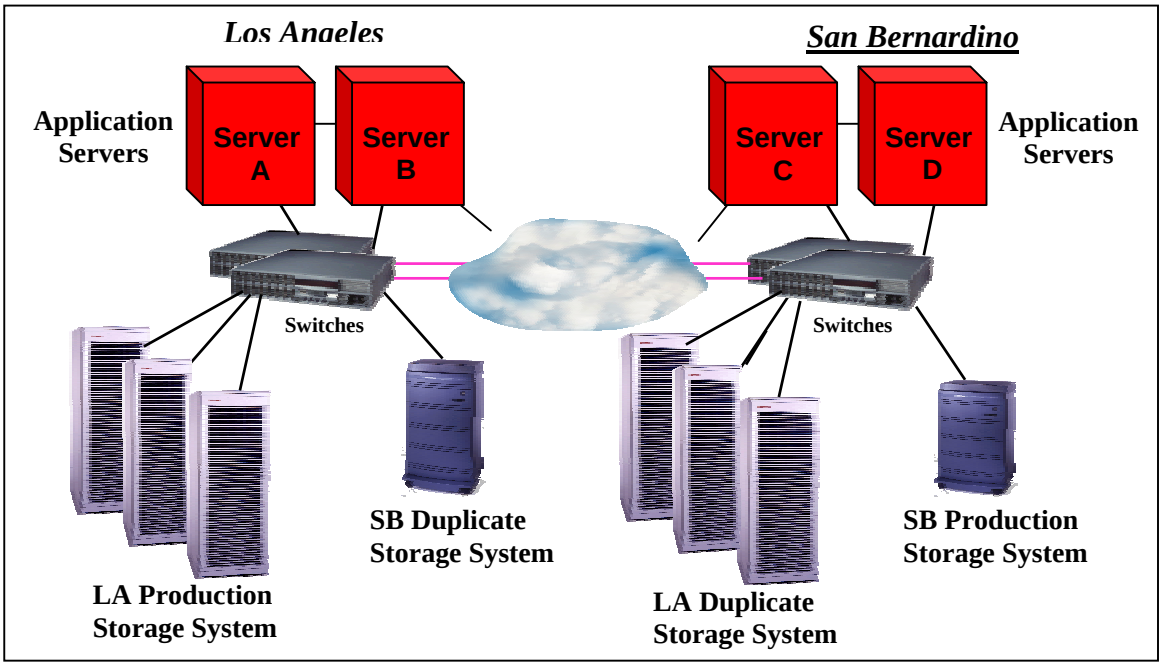
But your CFO strenuously objects to having a group of systems in San Bernardino just sitting idle, so you split your workload and give half of it to Los Angeles and half of it to San Bernardino. Notice that this is a multi-system-view implementation, because the target volume of a Continuous Access copy is not available for mounting by the systems. So, just as you duplicated the Los Angeles storage in San Bernardino, now you duplicate the San Bernardino storage in Los Angeles, which you then connect to the systems in Los Angeles as well, and you setup a Continuous Access copy from San Bernardino to Los Angeles.

So now you have your Los Angeles production data being replicated to San Bernardino, and your San Bernardino production data being replicated to Los Angeles. You could survive the loss of either data center, and have all of your data in the other one.

This is known as active/active bi-directional data replication. Even though each set of storage is only being replicated in one direction, your business has data being replicated across the enterprise.

Notice that the replication is being done by the FibreChannel. The hosts don't know or care that it is happening.

Failover requires you to explicitly stop the replication process in the surviving datacenter, and then explicitly mount the storage subsystems on the systems in the surviving datacenter, to get back into production. Failback requires the same type of operation, where you have to synchronize the data between the two storage subsystems, and then place one of them back into source mode and the other into target mode, in order to restore the standard configuration.



Notice that in both cases, system-based data high availability and StorageWorks Continuous Access, the data being replicated is actually being written multiple times, whether by the system or

by the storage controller. And all of the data on the source disk is being written to the target disk, whether it is mission-critical database files or temporary files in a scratch area. This requires careful planning to ensure that you are replicating everything you need (such as the startup scripts for the database application, which exists outside the database files and is probably not on the same disk volumes as the database files), but not too much (such as /tmp).

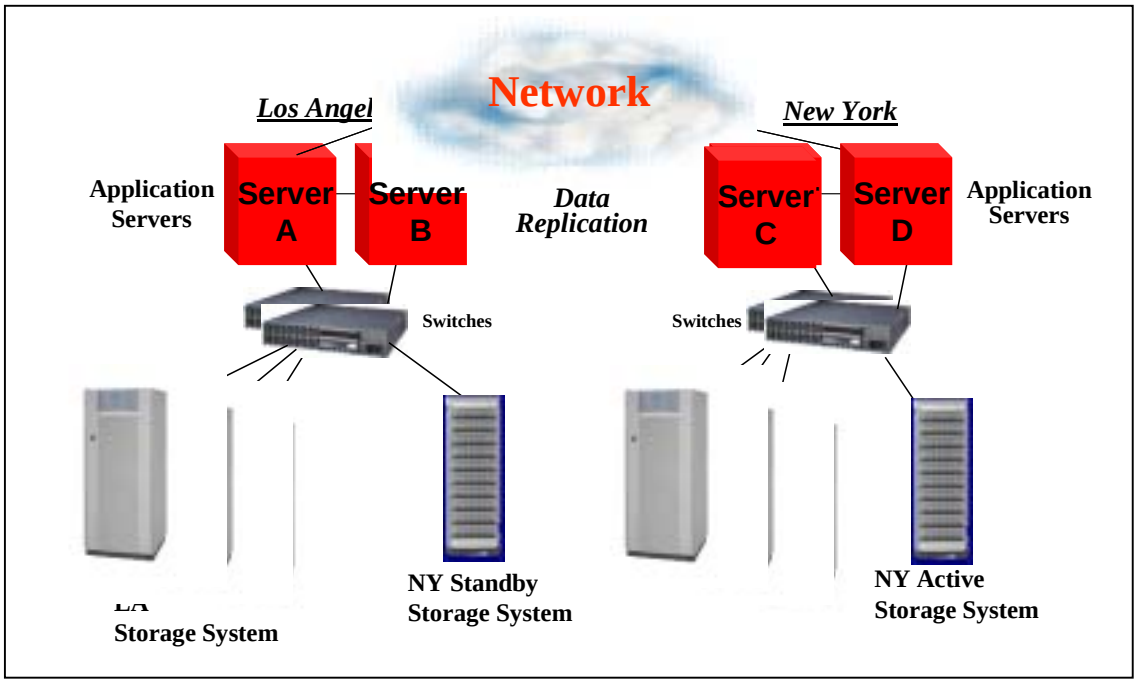
So how is this different from logical replication? Well, the biggest difference is what is being replicated and how the replication is being done.

Logical replication ignores the disk volumes and replicates the transactions themselves. In the same way that physical replication takes a single write operation and applies it to multiple disk volumes, logical replication takes a single update transaction and applies it to multiple databases. The communications can be done over standard networking, and the systems that operate the multiple databases may or may not be clustered, depending on the database software chosen.

Once again, you have your data center in Los Angeles, and it is working fine. Once again you decide you need a disaster recovery site. But in this case, you can put that disaster recovery site anywhere you want, because we are using standard networking technology. So you choose New York for your disaster recovery site.

You put a duplicate of your data center in New York, and then connect them with a fast networking link. Notice that this is a fully functional duplication of the data center, as it requires both systems and storage. However, it is not required to be a physical duplication: if you only require that some data is replicated, or if you can accept some lesser performance when a failover occurs, you can have fewer computing resources in New York than in Los Angeles. Then you use logical replication to replicate the database records.

You can either leave it like this, in active/standby mode, or you can have the New York site start to run some type of production. Notice that it has to be different from the production run in Los Angeles, because these are two different systems or clusters and cannot see the same data. But in the same way we had bi-directional physical replication, you can have bi-directional logical replication. This provides both data protection and failover capabilities from New York to Los Angeles. This is an active/active logical replication scheme. But keep in mind, just like in the previous example, failover and failback are semi-automated processes, with some human intervention required.



Keep in mind what is being replicated here. In the previous example of physical replication, the storage subsystem was taking whatever bits were written to the disk and copying them across to the other side. The storage subsystem doesn't have a clue about file systems, files, database records, re-do logs or transactions. The advantage of this is that *everything* is being replicated: database files, log files, flat files, everything. But the disadvantage is that a simple operator error (like "rm –r *" or "DELETE [*...]*.*;*" will be replicated to the target disk in real time.

But logical replication is replicating database transactions, not volumes. It does not replicate things like security information, patches to the operating system or applications, scripts, or any of the myriad of other files which are needed to maintain the site, all of which have to be replicated and maintained by hand. But the advantage is that an operator error that works on files cannot destroy your entire environment.

One last point about the differences between physical and logical replication. Physical replication does not understand the underlying structure of the data that it works on: it understands which disk blocks have changed, but not which files those blocks belong to or whether the disk block that was changed was a database row or a database index. Therefore, it is impossible to ensure atomicity of a transaction with physical replication. That is, there is a window of time where the write operation of (for example) the row information was successfully replicated but the index information has not yet been replicated. If the systems that originated the transaction fail, or the communications link fails at that moment, the replicated database is probably corrupt.

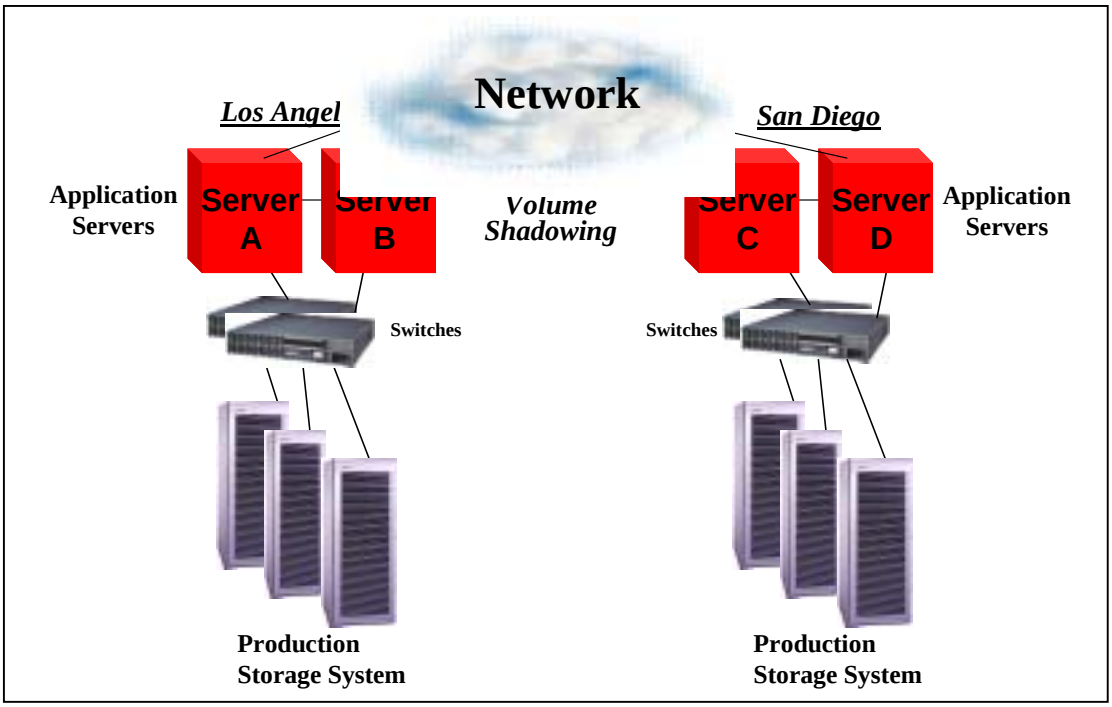
So, how is this different with OpenVMS Cluster Software and Disaster Tolerant Cluster Services (DTCS)? Well, one difference is where the physical replication is being done, but a more important difference is that the target volumes are fully accessible from any member of the wide area cluster.

Once again, you have your data center in Los Angeles, and it is working fine. Once again you decide you need a disaster recovery site. But in this case, you can put that disaster recovery site up to 800 km (500 miles) away, because we are using standard networking technology. So you choose San Diego for your disaster recovery site.

You put a duplicate of your data center down in San Diego, and then connect the two sites with an ATM fabric. There are lots of rules around this concerning latency and throughput, and quorum schemes get really interesting, so I really recommend you allow HP to help you design and implement it. But we have lots of sites working with this; it isn't that difficult.

Then you use host-based Volume Shadowing for OpenVMS to replicate the data. Because HBVS is host-based, all of the systems can mount the volumes directly, with full read/write access to the drives using direct access I/O. Any writes made by any of the systems will be written to the storage systems in both Los Angeles and San Diego, and all other systems will be aware of them with full cache coherency. Any reads will come from the local disks, so it is very efficient.

There are no standby systems here and no passive storage arrays. We recommend that the ATM fabric be redundant, so you are protected even in the event of network failure. But even if one of the sites was completely wiped out, the other site would recover very quickly, because the other systems already have the disks mounted and operational. It would simply be a cluster transition, and you are back in business.



So, why is there a distance limitation for this scenario, but not for extended clusters or storage based Continuous Access? Because this scenario requires a lot of two-way communication between the sites, and because of the speed of light.

Light travels in a vacuum at 186,282 miles per second. Let's round that off to 200,000 miles per second to make the math easy. The speed of 200,000 miles per second means light travels 200 miles per milli-second. We are worried about round-trip distances, so light can travel up to 100 miles away and back in 1 milli-second. But light travels somewhat slower in fibre than in a vacuum, and there are the inevitable switch delays, so the rule of thumb is that it takes1 milli-second for every 50-mile round trip. So 500 miles adds 10*1 = 10 milli-seconds to the latency of a disk access. Given normal disk access latency of 10-15 milli-seconds, this merely doubles the latency, and the OpenVMS Host-Based Volume Shadowing software can cope with that. But if you get much more than that, the software might think the disk at the other end has gone off-line, and the software will incorrectly break the shadow set. If we increase the timeout feature to get around this, the software will think the disk is just being slow when it really is inaccessible, and we will have unacceptable delays in breaking the shadow set when it needs to be broken.

In the physical and logical replication scenarios, each volume on one side is merely sending data to the other side, and eventually getting an acknowledgement back. The acknowledgement is not overly time critical, and you can keep sending new data while waiting for the acknowledgement of the data you have already sent. The other side cannot write data to the target disk, so there is no conflict resolution necessary. So the additional latency of the speed of light is not as important, so the distance limitations are almost non-existent, as long as you don't exceed the distance limitations of your interconnect technology.

Summary

For obvious reasons, every operating system offers a high availability option. But their capabilities vary widely: some systems offer 2-nodes with manual failover measured in minutes, other systems

offer 16 nodes with automated failover time measured in seconds, while still others offer hundreds or thousands of processing units with absolutely transparent recovery from failure. And each system knows how to protect itself in this increasingly insecure world: some systems do it by depending on FibreChannel replication, others do it by depending on a database to move the data around the country, and others offer true active/active multi-site clusters over hundreds of miles.

It is up to you as technologists to understand these technologies, and to pick the right one for the business task. But you have an additional job that you might not think of: letting your senior management know the exact capabilities of the technology you chose. Because if you implemented a 2-node, manual failover system with no disaster tolerance, and your management thinks you have implemented an infinitely expandable fault tolerant system with zero recovery time even if the primary datacenter is completely and unexpectedly wiped out, you have a problem. And you will only discover you have a problem when the system behaves exactly as you implemented it, and management discovers that it didn't do what they thought it would do.

So the answer is to document exactly what your chosen solution will and won't do, and get full agreement from your senior management that this is what they want it to do. And if they come back and say they need it to do more, then you request more money to get it to do what they need. In either scenario, HP is available to provide the software and services to help you design and implement the chosen solution.

Acknowledgements

I wish to thank Kirk Bresniker, Dan Cox, Jon Harms, Keith Parris, Bob Sauers and Wesley Sawyer for their invaluable input and review of this material.

For More Information

On LifeKeeper for Linux

- http://www.hp.com/linux for general information on HP servers and Linux
- http://h18000.www1.hp.com/solutions/enterprise/highavailability/linux/index.html for general information on HP servers and Linux and high availability solutions
- http://www.hp.com/hpinfo/newsroom/press/08aug02b.htm for information on Lustre
- http://www.compaq.com/Solutions/enterprise/highavailability/linux/description.html#specs for specifications on LifeKeeper with ProLiant servers
- http://linux-ha.org
    - "What Linux-HA Can Do Now"
    - "LAN Mirroring Technologies"
- http://www.missioncriticallinux.com/technology/cluster/ for the white paper on Mission Critical Linux
- http://technet.oracle.com/tech/linux/ for information on Oracle and Linux
- http://www.steeleye.com/products/linux/#2 and http://www.steeleye.com/pdf/literature/lkpr4linux.pdf for information on SteelEye LifeKeeper
- http://www.kernel.org/software/mon/ for information on the Service Monitoring Daemon
- http://www.tildeslash.com/monit/ for information on the Monit Utility

On Serviceguard for HP-UX and Linux

- http://www.hp.com/go/ha for general information on Serviceguard and high availability
- http://www.hp.com/products1/unix/operating/infolibrary/reports/2002Unix_report.pdf for the DH Brown 2002 UNIX Function Review report
- http://docs.hp.com/hpux/onlinedocs/ha/highly_avail_clust.pdf for Disaster Tolerant and Highly Available Cluster Technologies
- http://www.hp.com/products1/unix/highavailability/ar/mcserviceguard/infolibrary/index.html, Information Library
    - 5nines Architecture Overview
    - System Cluster Technologies and Disaster Tolerance
    - Data Protection
- http://www.software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?productNumber=B2491BA for MirrorDisk/UX

On NonStop Kernel

- http://nonstop.compaq.com/view.asp?PAGE=TIM_Prod for access to the Total Information Manager (TIM) product, for full NSK information
- http://h71033.www7.hp.com/object/tdnsk3pd.html for details on NSK and high availability
- http://h71033.www7.hp.com/page/RDF_SW.html for information on Remote DataCenter Facility

On TruCluster

- http://h30097.www3.hp.com/ for general information on Tru64 UNIX and TruCluster
- http://h18000.www1.hp.com/products/quickspecs/11444_div/11444_div.HTML for the QuickSpecs on TruCluster V5.1b
- http://h30097.www3.hp.com/docs/pub_page/cluster51B_list.html for the documentation. Specifically:
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGVETE/TITLE.HTM , Cluster Technical Overview
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGVETE/TITLE.HTM, Cluster Technical Overview, Section 2.2
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGVETE/TITLE.HTM, Cluster Technical Overview, Section 3
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGWETE/TITLE.HTM, Hardware Configuration, Section 1.3.1.4
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGYETE/TITLE.HTM, Cluster Administration, Section 4.3, Calculating Cluster Quorum
    - http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHH0ETE/TITLE.HTM, Highly Available Applications, Chapter 1, Cluster Applications

- http://h18000.www1.hp.com/products/quickspecs/10899_div/10899_div.HTML for the QuickSpecs for the Logical Storage Manager V5.1b
- http://www.hp.com/techservers/ for the AlphaServer SC home page

On OpenVMS Cluster Software

- http://h71000.www7.hp.com/ for general information on OpenVMS and OpenVMS Cluster Software
- http://h71000.www7.hp.com/openvms/products/clusters/index.html for information on OpenVMS Cluster Software V7.3-1
- http://h18000.www1.hp.com/info/SP2978/SP2978PF.PDF for OpenVMS Cluster Software V7.3-1 SPD
- http://h71000.www7.hp.com/doc/index.html for the documentation. Specifically:
  - http://h71000.www7.hp.com/doc/731FINAL/4477/4477PRO.HTML, OpenVMS Cluster Systems, Section 2.3.2 shows quorum algorithm, and Chapter 7, Setting Up and Managing Cluster Queues
  - http://h71000.www7.hp.com/doc/731FINAL/6318/6318PRO.HTML, Guidelines for OpenVMS Cluster Configurations, Chapter 8, Configuring OpenVMS Clusters for High Availability and Appendix D, Multi-Site OpenVMS Clusters
  - http://h71000.www7.hp.com/doc/731FINAL/5423/5423PRO.HTML, Volume Shadowing for OpenVMS, Section 1.5 discusses WAN based RAID-1 for disaster tolerance

On Windows 2000

- http://www.microsoft.com/windows2000/en/datacenter/help/ for general Windows 2000 DataCenter information
- http://www.microsoft.com/windows2000/en/datacenter/help/ for the documentation. Specifically:
  - Choosing a Cluster Model, emphasizes that Windows 2000 DataCenter is a multi-system-view cluster, and acknowledges the lack of single-system-view capabilities.
  - Checklist: Preparing a Server Cluster, states that each system in the cluster must have its own system disk.
  - Server Clusters says up to 4 systems in a server cluster
  - Cluster Hardware and Drivers says the network is the only cluster interconnect
  - Quorum Disk describes the use of the quorum disk, and Cluster Database discusses how the cluster database from each system is written to the recovery log on the quorum disk
  - Windows Clustering, Server Clusters, How To…, Perform Advanced Administrative Tasks
  - Choosing a RAID Method
  - Disaster Protection discussing the lack of WAN RAID

On Single System Image Linux

- http://sourceforge.net/projects/ssic-linux for information on this HP project

On StorageWorks Continuous Access

- http://h18006.www1.hp.com/storage/software.html for general information on StorageWorks high availability solutions
- http://www.compaq.com/products/quickspecs/10281_na/10281_na.html, QuickSpecs for StorageWorks Continuous Access
- http://h18006.www1.hp.com/products/storage/software/conaccesseva/index.html, SANworks Continuous Access Overview and Features

Books

- "Clusters for High Availability", Peter Weygant, ISBN 0-13-089355-2
- "In Search of Clusters", Gregory F. Pfister, ISBN 0-13-899709-8