OpenVMS Technical Journal

V5, February 2005

Table of Contents

OpenVMS Technical Journal V5

A Survey of Cluster Technologies

A Survey of Cluster Technologies

Ken Moreau, Solutions Architect, OpenVMS Ambassador

Overview

This paper surveys the cluster technologies for the operating systems available from many vendors, including IBM AIX, HP HP-UX, Linux, HP NonStop Kernel, HP OpenVMS, PolyServe Matrix Server, Sun Microsystems Solaris, HP Tru64 UNIX and Microsoft Windows 2000/2003. In addition, it discusses some technologies that operate on multiple platforms, including MySQL Cluster, Oracle 9i and 10g Real Application Clusters, and Veritas clustering products. It describes the common functions that all of the cluster technologies perform, shows where they are the same and where they are different on each platform, and introduces a method of fairly evaluating the technologies to match them to business requirements. As much as possible, it does not discuss performance, base functionality of the operating systems, or system hardware.

The focus for the audience of this document is a person who is technically familiar with one or more of the clustering products discussed here and who wishes to learn about one or more of the other clustering products, as well as anyone who is evaluating various cluster products to find which ones fit a stated business need.

Introduction

Clustering technologies are highly interrelated, with almost everything affecting everything else. This subject is broken down into five areas:

- Single/multisystem 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 this article will describe how they work, in detail.
- Configurations, which defines how you assemble a cluster, both physically and logically.
- Application support, which discusses how applications 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 to a clustered environment?
- Resilience, which describes what happens when bad things happen to good computer rooms. This covers host-based RAID, wide area "stretch" clusters, extended clusters, and disaster tolerant scenarios.

This article covers the capabilities of IBM High Availability Cluster Multiprocessing (HACMP) 5.1 for AIX 5L, Linux LifeKeeper V4.3, Microsoft SQL Server 2000 Enterprise Edition, MySQL Cluster 4.1, HP NonStop Kernel G06.22, HP OpenVMS Cluster Software V7.3-2, Oracle 9i/10g Real Application Clusters, PolyServe Matrix Server and Matrix HA for Linux and Windows, HP Serviceguard 11i for HP-UX, HP Serviceguard A.11.16 for Linux, Sun Microsystems SunCluster 3.1 in a SunPlex cluster of Solaris 9 servers, HP TruCluster V5.1b, Veritas Cluster Server and SANPoint Foundation Suite V3.5 and Windows 2000/2003 with the Cluster Service. It also discusses Windows SQL Server 2005 Enterprise Edition, which offers additional capabilities beyond SQL Server 2000. This has not been officially released by Microsoft at this time, but is close enough that it is safe to describe its functionality.

For Linux, the focus is on the high availability side, not the HPTC (i.e., Beowulf) technologies.

Single-System and Multisystem-View Clusters

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

- Availability defines whether the application stays up, even when components of the cluster go down. If you 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 the 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. Any actions that the users are forced to take as part of this failover, such as logging in to the new

system, must be considered as part of the failover time, because the users are not doing productive work during that time. Further, if the application is forced to pause on the surviving system during the failover, to the users on the second system the application is down for those 30 seconds.

- Reliability defines how well the system performs during a failure of some of the components. If you get subsecond query response and if a batch job finishes in 8 hours with all of the systems in the cluster working properly, do you still get that level of performance if one or more of the systems in the cluster is down? If you 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 the users neither know nor care how many systems there are in the cluster; they only care whether they can rely on the environment to get their work done.

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

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

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

Multisystem-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. Applications cannot simultaneously access data from both systems at the same time, and the operating system files cannot be shared between the two systems. Therefore, 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 is required.

Multisystem-View Clusters In Active-Passive Mode

Multisystem-view clusters in 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. It is called active-passive because during normal operations, one of the systems is active and the other is not . This is classically called N+1 clustering, where N=1 for a two-system cluster. For clusters with larger numbers of systems, one or more spare servers can take over for any of the active systems.

Failover can be manual or automatic. Because the systems are all cabled to the same storage array, a spare system monitors the primary system and starts the services on the spare system if it detects a failure of the primary system. The "heartbeat" function can come over the network or from a private interface.

Comparing a single system to a multisystem-view cluster in an active-passive mode, the availability, reliability, scalability, and manageability characteristics are as follows:

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

- Reliability can be nearly perfect in this environment, because if all of the 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, 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 two systems as it does to manage a single system. There are multiple system roots, so any patches or other updates need to be installed multiple times, backups need to be done multiple times, and so

forth. Furthermore, you have to test the failover and failback, which adds to the system management workload

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

Multisystem-View Clusters in Active-Active Mode

The physical environment of a multisystem-view cluster in active-active mode is identical to that of active-passive mode. Two or more systems are physically cabled to a common set of storage, but only able to mount each file system on one of the systems. The difference is that multiple systems 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 any files between the systems.

For example, a database environment could segment their customers into two groups, such as by the first letter of the last name (for example, A-M and N-Z). Then each group would be set up on separate disk volumes 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 including the spare, they really have five tires. They are operating in an N+1 environment, because four tires are required for minimum operation of the vehicle. A variation is to use the equivalent of the "donut" tire – a server that offers limited performance but enough to get by for a short period of time. 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 come over the network or from a private interface. Comparing a single system to a multisystem-view cluster in active-active mode, the availability, reliability, scalability, and manageability characteristics are as follows:

- Availability is increased because you now have multiple systems available to do the work. As in the active-passive environment, the odds of all systems being broken at the same time are fairly low but still present.
- Reliability may be increased, but is commonly decreased. If two systems are each running at 60% of capacity, the failure of one will force the surviving system to work at 120% of capacity, which is not optimum because you should never exceed about 80% of capacity.
- Scalability for any given workload 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 Multisystem-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 multisystem-view cluster, whether active-active or active-passive. The surviving system must:

- Notice that the other system is no longer available, which is detected 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 part of the definition of multisystem-view clusters. The surviving system must mount the disks that were mounted on the other system, and then possibly perform consistency checking on each volume. If you have a large number of disks, or large RAID sets, this could take a long time.
- Start the applications that were active on the failing system.
- Initiate the recovery sequence for that software. For databases, this might include processing the journalling logs in order to process any in-flight transactions that the failing system was performing at the time of the failure.

In large environments, it is not unusual for this operation take 30-60 minutes. 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 not running at full speed, 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 shared storage and can directly mount all shared 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 may be spare capacity, but there are no spare systems. All systems can run all applications at all times.

In a single-system-view cluster, there can be many systems. Comparing a series of independent systems to the same number of systems in a single-system-view cluster, the availability, reliability, scalability, and manageability characteristics are 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 potentially you can have many systems in the cluster.
- Reliability is much better, because with many systems in the cluster, the workload of a single failed system can 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, 1/3 of the load is placed 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. If you have an application that 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. There is no increase in management workload even when you have many systems.

The advantages in failover times over multisystem-view clusters comes from not having to do quite so much work during a failover:

- The surviving systems must detect the failure of the system. This is common between the two types of clusters.
- 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, but it can begin almost instantly in the single-system-view cluster case. The application recovery time will be similar on both types of clusters, but if you have a large number of small systems, you can achieve parallelism even in recovery, so that your recovery can be faster in this case as well.

One criticism of shared root environments with a single root for the entire cluster is that this represents a single point of failure. If a hardware failure causes the shared root device to be inaccessible, or an operator error causes corruption on the shared root (such as applying a patch incorrectly or deleting the wrong files), the entire cluster will be affected. These concerns must be balanced against the amount of work involved in maintaining multiple system roots. Furthermore, an incorrect patch on one system root can cause incompatibility with the other cluster members. Such problems can be difficult to diagnose.

The system administrator must set up the operational procedures (including the number of shared roots) for their environment in such as way that the possibility of failure is minimized, and services are still delivered in a cost-effective manner. Frequent backups, hardware and software RAID, and good

quality assurance and testing procedures can help reduce the possibility of failure in either environment.

Now that the terms are defined, you can see how different cluster products work.

	Multisystem view	**Single-system view**	**Shared root**
HACMP AIX, Linux	Yes	No	No
LifeKeeper Linux, Windows	Yes	No	No
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	Yes (MySQL Server)	Yes	No
NonStop Kernel	Yes	Yes	Each node (16 CPUs)
OpenVMS Cluster Software OpenVMS	Yes	Yes	Yes
Oracle 9i/10g RAC Many O/S's	Yes (Oracle DB)	Yes	Effectively yes ($ORACLE_HOME)
PolyServe Matrix Linux, Windows	Yes	Yes	No
Serviceguard HP-UX, Linux	Yes	No	No
SQL Server 2000/2005 Windows	Yes	No	No
SunCluster Solaris	Yes	No	No
TruCluster Tru64 UNIX	No	Yes	Yes
Veritas Cluster Server AIX, HP-UX, Linux, Solaris, Windows	Yes	No	No
Windows 2000/2003 Cluster Service Windows	Yes	No	No

Figure 1 Types of Clusters

HACMP

High Availability Cluster Multiprocessing (HACMP) 5.1 for AIX 5L runs on the IBM pSeries (actually an RS/6000 using the Power4 chip), and for Linux runs on a variety of platforms. It is a multisystem image cluster, where each system in the cluster requires its own system disk. Management is done either through the included Cluster Single Point Of Control (C-SPOC) or by the layered product Cluster Systems Management (CSM), which can manage mixed AIX and Linux systems in the same cluster. In both cases, you issue commands one time and they are propagated to the different systems in the cluster. Clusters can be configured either as active-passive (which IBM calls "standby") or active-active (which IBM calls "takeover") configurations.

Previous versions of HACMP came in two varieties: HACMP/ES (Enhanced Scalability) and HACMP (Classic). V5.1 includes all of the features of HACMP/ES.

Linux Clustering

Linux clustering is focused either on massive system compute farms (Beowulf and others) or a high availability clustering scheme. This article specifically does not address the High Performance Technical Computing market here, which breaks down a massive problem into many (hundreds or thousands) tiny problems and hands them off to many (hundreds or thousands) small compute engines. This is not really a high availability environment because if any of those compute engines fails, that piece of the job has to be restarted from scratch.

Most of the Linux high availability efforts are focused on multisystem-view clusters consisting of a small number of systems from which applications can fail over from one system to the other. Cluster file system projects such as Lustre and GFS are discussed later, but these do not offer shared root, so systems in Linux clusters require individual system disks.

There are some other projects that are focused on single-system-view clusters. One of these is the work being done by HP as part of the Single System Image Linux project. Another is from Qlusters Corporation, specifically the ClusterFrame XHA and ClusterFrame SSI products based on OpenMosix. At this time these are focused on the HPTC market, but when they prove themselves in the commercial high availability market space, they will have significant capabilities that match or exceed every other clustering product. Visit http://openssi.org for more information on the HP project, and http://www.qlusters.com for more information on ClusterFrame XHA and SSI.

MySQL Cluster

MySQL Cluster is a layer on top of MySQL, the open source database that runs on AIX, HP-UX, Linux (Red Hat and SUSE), Mac OS X, Windows 2000 and XP, and is being planned for OpenVMS. The software and intellectual property were acquired from Ericsson, and was integrated as open source into the Storage Engine of MySQL Server.

There are three components to a MySQL Cluster: application nodes, database server or storage nodes, and management nodes. Application nodes run MySQL Server and connect to the database server nodes running MySQL Cluster, and are managed by the management nodes. The different nodes can either be processes on a single server or distributed on multiple servers. MySQL Cluster is designed to work on "shared nothing" operating systems, where each node has private storage.

MySQL offers a multisystem view of the database, and MySQL Cluster adds single-system view. It does not support sharing of disks, but transparently fragments the database over the systems in the cluster with real-time replication, so that the database information can be accessed from any system in the cluster.

NonStop Kernel

NonStop Kernel (NSK, formerly the Tandem Guardian operating system) runs on the HP NonStop servers (formerly NonStop Himalaya or Tandem Himalaya servers), and is configured as a single-system-view cluster. It offers true linear scalability as you add processors to the environment, because of the shared-nothing architecture and superb cluster interconnect. 2 to 16 processors can be configured to have a shared root and be considered one system. A cluster of systems, both local and geographically distributed, is centrally managed with the Open Systems Manager (OSM) console.

OpenVMS Cluster Software

OpenVMS Cluster Software has always been the gold standard of clustering, with almost linear scalability as you add systems to the cluster. It can be configured as either multisystem view or single-system view, although the most common is single-system view. It supports single or multiple system disks.

Oracle 9i/10g Real Application Clusters

Oracle 9i/10g Real Application Clusters (RAC) is the next generation of Oracle Parallel Server, and runs on the Oracle 9i and 10g database on every major computing platform. It offers a single-system-view of the database files, such that external applications can connect to the database instance on any of the systems in the cluster. It does not offer a multisystem-view of the database, but this is easily achieved by simply running the database without RAC.

Oracle achieves the functionality of a shared root (called $ORACLE_HOME), but accomplishes it differently on the different host operating systems. On single-system-view operating systems that offer

clustered file systems, $ORACLE_HOME is placed in a shared volume and made available to all of the systems in the cluster. On multisystem-view operating systems that do not offer clustered file systems, Oracle replicates all of the operations to individual volumes, one per system in the cluster, without forcing the user to take any action. The installation, patches, and monitoring are the same whether there is one $ORACLE_HOME or multiple, replicated $ORACLE_HOMEs.

Oracle is steadily adding functionality to RAC, which requires less support from the base operating systems. For example, 9i RAC required the addition of Serviceguard Extensions for RAC (SGeRAC) on HP-UX, while 10g RAC does not require SGeRAC. Further, 10g RAC is capable of running without the underlying operating system itself being clustered. As a result, HACMP, Serviceguard, SunClusters, and Windows 2000/2003 Cluster Server are now optional for Oracle 10g RAC.

PolyServe Matrix HA and Matrix Server

PolyServe Matrix Server is a clustered file system for Linux and Windows which includes a high availability and a cluster management component. The HA component provides automated failover and failback of applications. Each node in a PolyServe cluster requires its own system disk, which can be local or SAN boot. Matrix Server allows the underlying disk volumes to be accessed for read-write simultaneously from all nodes. It also allows a unified view of device management, such that device names are common across all systems in the cluster regardless of the order that the devices were discovered during a boot. The management application is CLI and GUI based, and allows the cluster to be managed as a single entity from any node in the cluster. Matrix Server is primarily an installable file system, and so does not itself offer a multisystem view because the underlying operating systems offer that as the default. Similarly, Matrix Server does not offer a shared root, because it is a layer on top of the operating system and is activated late in the boot process.

Serviceguard

Serviceguard (also known as MC/Serviceguard) is a multisystem-view failover cluster. Each system in a Serviceguard cluster requires its own system disk. There are excellent system management capabilities from 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, and install new instances of the operating system and applications by copying existing instances using Ignite/UX. It is also well integrated with HP/OpenView.

Serviceguard Manager can configure, administer, and manage HP-UX and Linux Serviceguard clusters through a single interface. Each cluster must be homogeneous; that is, each cluster can only be running one operating system. Business continuity solutions to achieve disaster tolerance are available. HP-UX offers Campuscluster, Metrocluster, and Continentalcluster. Metrocluster functionality is offered on Linux through Serviceguard for Linux integration with Cluster Extension XP. Additional complementary products on Linux include Serviceguard Extension for SAP for Linux and an application toolkit for Oracle. Contributed toolkits are available for leading Linux applications.

SQL Server 2000/2005 Enterprise Edition

Microsoft SQL Server 2000 Enterprise Edition is a multisystem-view failover clustered database, running on Microsoft Windows 2000/2003. SQL Server 2005 is the next release of this product, and is available on Windows 2003. They are available in both 32-bit and 64-bit versions for the various hardware platforms. They provide both manual and automatic failover of database connections between servers. A database can be active on only a single instance of SQL Server, and each server requires its own installation. Unless specifically noted, all references to functionality in this article apply to both versions equally.

SunCluster

SunCluster 3.1 is a multisystem-view failover cluster. A group of Solaris servers running SunCluster software is called a SunPlex system. Each system in a SunPlex requires its own system disk, and Sun recommends keeping the "root" passwords the same on all systems. This has to be done manually, which gives you some idea about the level of management required by a SunPlex. The Cluster File System (CFS) offers a single-system-view of those file systems that are mounted as a CFS. The Sun Management Center and SunPlex Manager are a set of tools that manage each system as a separate entity but from a centralized location.

TruCluster V5.1b

TruCluster V5.1b represents a major advance in UNIX clustering technology. It can only be configured as a single-system-view. The clustering focus is on managing 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 and a single copy of almost all system files.

Veritas

Veritas offers several products in this area, but then offers many combinations of these products under separate names. The base products are:

- Veritas Cluster Server (VCS) manages systems in a cluster, with a GUI interface. It is unique in that it can manage multiple different clusters at a time. It can simultaneously manage systems running AIX, HP-UX, Linux, Solaris, and Windows running the Veritas Cluster Server software. Each cluster must be homogeneous; that is, each cluster can only be running one operating system.
- Veritas File System (VxFS) is a journaled file system that works in either a standalone system or a cluster. A "light" version of this product is included with HP-UX 11i Foundation Operating Environment, and the full version is included in the Enterprise Operating Environment as Online JFS.
- Veritas Global Cluster Manager (GCM) manages geographically-distributed Veritas Cluster Server clusters from a central console. Applications can be monitored across multiple clusters at multiple sites, and can be migrated from one site to another. The application service groups in each cluster must be setup by VCS, but can then be monitored and migrated through GCM.
- Veritas Volume Manager (VxVM) manages volumes, whether they are file systems or raw devices. A "light" version of this product is included with HP-UX 11i, and offers similar functionality as the HP-UX 11i Logical Volume Manager and the TruCluster Logical Storage Manager.
- Veritas Cluster Volume Manager (CVM) offers the same functionality as VxVM but does it across multiple systems in a cluster. An important distinction is that CVM requires that every system mount every shared volume.
- Veritas Volume Replicator (VVR) allows disk volumes to be dynamically replicated by the host, both locally and remotely. This is similar to the "snap/clone" technology in the StorageWorks storage controllers.

Veritas combines these into many different packages. The two important ones for this discussion are:

- SANPoint Foundation Suite – HA (SPFS – HA), which includes VxFS with cluster file system extensions, VxVM with cluster extensions, and the Veritas Cluster Server
- Veritas DataBase Extension Advanced Cluster (DBE/AC) for Oracle 9i/10g Real Application Clusters (RAC), which includes VxFS, VxVM, and CVM, along with an implementation of the Oracle Disk Manager (ODM) API for Oracle to use to manage the volumes

The Veritas Network Backup (NBU) is not a cluster technology; therefore, it is not addressed in this paper.

Most of these products run under AIX, HP-UX, Linux, Solaris, and Windows, but SANPoint Foundation Suite – HA runs only under HP-UX and Solaris. Check with Veritas for specific versions and capabilities of the software for specific versions of the operating systems, and look for more discussion of these in later sections of this paper. In some cases the products replace the operating system's clusterware (Cluster Server, Cluster Volume Manager), and in other cases they are enhancements to the operating system's products (Cluster File System, Volume Replicator). All of the products are offered by both HP and Veritas, and supported by either company through the Cooperative Service Agreement (ISSA).

Windows 2000/2003 Cluster Service

Windows 2000/2003 DataCenter is a multisystem-view failover cluster. Applications are written to fail over from one system to another. Each system in a Windows 2000/2003 cluster requires its own system disk, but the Cluster Administrator tool can centralize the management of the cluster.

Cluster File Systems

Cluster file systems are how systems communicate with the storage subsystem in the cluster. There are really two technologies here: one addresses how a group of systems communicates with volumes that are physically connected to all of the systems, and the other addresses 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 in most implementations. 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. If any other system in the cluster wants to access files on the volume, 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:

1. An I/O connection is initiated across the cluster interconnect from system B to system A.
2. System A receives the request, and initiates the I/O request to the volume.
3. System A gets the data back from the volume, and then sends an I/O request 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 in an active-active system.

Every system offers network I/O in order 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. This type of access is known as "proxy file system."

In contrast, direct access I/O (also known as "concurrent I/O") allows each system to independently access any and all devices, without going through any other node in the cluster. Notice that this is different from UNIX direct I/O, which simply bypasses the file system's 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 system's cache.

Implementing direct access I/O allows a cluster file system to eliminate two of the 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.

You may object that we could overwhelm a single disk with too many requests. This is 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.

These technologies are focused on the commercial database environments. But in the High Performance Technical Computing (HPTC) environment, the requirements are slightly different. The IBM General Parallel File System (GPFS) offers direct access I/O to a shared file system, but focuses on the HPTC model of shared files, which differs from the commercial database model of shared files in the following ways:

- The commercial model optimizes for a small number of multiple simultaneous writers to the same area (byte range, record or database row) of a shared file, but assumes that this occurs extremely frequently, because commercial databases and applications require this functionality.
- The HPTC model optimizes for throughput because, while the number of multiple simultaneous writers to any given file may be large (hundreds or even thousands of systems), the applications are designed so that only one process is writing to any given byte range. In the unlikely event of multiple writers to a single byte range of a shared file, the HPTC model switches to network I/O semantics, and ships all of the data to a single master system for that

byte range. This has been found to be more efficient overall because the condition occurs so infrequently in the HPTC world.

This paper focuses on the commercial database environment.

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

	Network I/O	**Direct Access I/O**	**Distributed Lock Manager**
HACMP AIX, Linux	Yes	Raw devices and GPFS	Yes (API only)
LifeKeeper Linux, Windows	NFS	Supplied by 3rd parties	Supplied by 3rd parties
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	No (supplied by native O/S)	Yes (effectively)	Yes (for database only)
NonStop Kernel	Data Access Manager	Effectively Yes	Not applicable
OpenVMS Cluster Software OpenVMS	Mass Storage Control Protocol	Files-11 on ODS-2 or -5	Yes
Oracle 9i/10g RAC Many O/S's	No (supplied by native O/S)	Yes, both raw devices & Oracle file systems	Yes
PolyServe Matrix Linux, Windows	No (supplied by native O/S)	Yes	Yes (for file system only)
Serviceguard HP-UX, Linux	Yes	Supplied by 3rd parties	Supplied by 3rd parties
SQL Server 2000/2005 Windows	No (supplied by the native O/S)	No	No
SunCluster Solaris	Yes	Supplied by 3rd parties	Supplied by 3rd parties
TruCluster Tru64 UNIX	Device Request Dispatcher	Cluster File System (requires O_DIRECTIO)	Yes
Veritas SPFS HP-UX, Solaris	No (supplied by the native O/S)	Yes (SPFS or DBE/AC)	Yes (SPFS or DBE/AC)
Windows 2000/2003 Cluster Service Windows	NTFS	Supplied by 3rd parties	Supplied by 3rd parties

Figure 2 Cluster I/O Attributes

Every system in the world can do network I/O in order to share devices that are on private storage busses.

HACMP, LifeKeeper, and Serviceguard do network I/O using NFS; NonStop Kernel does it with the Data Access Manager (DAM, also called the "disk process" or DP2); OpenVMS Cluster Software does it with the Mass Storage Control Protocol (MSCP); SunCluster does it with NFS or the Cluster File System; TruCluster does it both with the Device Request Dispatcher (DRD) and the Cluster File System; and Windows 2000/2003 does it with NTFS and Storage Groups. MySQL, Oracle, SQL Services, and Veritas use the native I/O system of the operating system on which they are running.

The more interesting case is direct access I/O.

HACMP offers direct access I/O to raw devices for two to eight systems in a cluster. However, HACMP does not itself handle the locks for raw devices. Instead, it requires that applications use the Cluster Lock Manager APIs to manage concurrent access to the raw devices. The Concurrent Logical Volume Manager provides "enhanced concurrent mode," which allows management of the raw devices through the cluster interconnect, which should not be confused with a cluster file system as it applies only to raw devices.

Linux has projects being done by HP and Cluster File Systems Inc for the US Department of Energy to enhance the Lustre File System originally developed at Carnegie Mellon University. This enhancement is focused on high-performance technical computing environments and is called the Scalable File Server. This uses Linux and Lustre to offer high throughput and high availability for storage, but does not expose this clustering to the clients.

MySQL Cluster does not offer direct access I/O, but it achieves the same effect by fragmenting the database across the systems in the cluster and allowing access to all data from any application node in the cluster. This provides a unified view of the database to any application that connects to the MySQL Cluster. Each database node is responsible for some section of the database, and when any data in that section is updated, the database nodes synchronously replicate the changed information to all other database nodes in the cluster. It is in fact a "fragmented" (using MySQL terminology) and a "federated" (using Microsoft and Oracle terminology) database, and yet it behaves as a single-system image database.

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

OpenVMS Cluster Software extends the semantics of the Files-11 file system transparently into the cluster world, offering direct I/O to any volume in the cluster from any system in the cluster that is physically connected to the volume. For volumes that are not physically connected to a specific system, OpenVMS Cluster Software transparently switches to network I/O. Opening a file for shared access by two processes on a single system, and opening the same file for shared access by two processes on two different systems in a cluster, works identically. In effect, all file operations are automatically cluster-aware.

Oracle 9i/10g RAC does not offer network I/O, but requires that any volume containing database files be shared among all systems in the cluster that connect to the shared database. 9i/10g RAC offers direct access I/O to raw devices on every major operating system, with the exception of OpenVMS, where it has used the native clustered file system for many years (starting with the original version of Oracle Parallel Server). Oracle has implemented its own Oracle Clustered File System (OCFS) for the database files on Linux and Windows as part of Oracle 9i RAC 9.2, and is extending OCFS to other operating systems in Oracle 10g as part of the Automated Storage Manager (ASM).

In general, the OCFS cannot be used for the Oracle software itself ($ORACLE_HOME), but can be used for the database files. The following table shows which cluster file systems can be used for Oracle software and database files:

	Oracle software	**Oracle database files**
HACMP/ES on AIX	Local file system only	Raw, GPFS
LifeKeeper on Linux	Local file system only	RAW, OCFS
OpenVMS Cluster SW	OpenVMS cluster file system	OpenVMS cluster file system
Serviceguard on HP-UX	Local file system only	RAW, Veritas DBE/AC
Serviceguard on Linux	Local file system only	Raw, OCFS
SunClusters on Solaris	Solaris GFS	Raw, Veritas DBE/AC

		(Solaris GFS is not supported for database files)
TruCluster on Tru64 UNIX	TruCluster CFS	Raw, TruCluster CFS
Windows 2000/2003 Cluster	OCFS	Raw, OCFS

Figure 3 Cluster File Systems for Oracle

"Local file system only" means that the Oracle software ($ORACLE_HOME) cannot be placed on a shared volume; each server requires its own copy, as described above. Interestingly, the Solaris Global File Service does support a shared $ORACLE_HOME, but does not support shared Oracle database files.

PolyServe Matrix Server does not offer network I/O as such, because it is available in the underlying operating systems. Matrix Server performs direct access I/O to any volume of the cluster file system under its control and uses its distributed lock manager to perform file locking and cache coherency. It supports on-line addition and removal of storage, and the meta-data is fully journaled. PolyServe Matrix Server and OpenVMS Cluster Software are the only cluster products with fully distributed I/O architectures, with no single master server for I/O. PolyServe manages the lock structure for each file system independently of the lock structures for any other file systems, so there is no bottleneck or single point of failure.

Serviceguard and Windows 2000/2003 Cluster Service do not offer a direct access I/O methodology of their own, but rely on 3rd party tools such as Oracle raw devices or the Veritas SANpoint Foundation Suite – High Availability. Serviceguard uses an extension to the standard Logical Volume Manager for clusters, called the Shared Logical Volume Manager (SLVM) to create volumes that are shared among all of the systems in the cluster. Notice that this only creates the volume groups: the access to the data on those volumes is the responsibility of the application or the 3rd party cluster file system.

SQL Services 2000/2005 does not offer direct access I/O.

SunCluster does not offer direct access I/O in its cluster file system (Global File Service, or GFS), which simply allows access to any device connected to any system in the cluster, independent of the actual path from one or more systems to the device. In this way devices on private busses such as tapes or CD-ROMs can be accessed transparently from any system in the SunPlex. The GFS is a proxy file system for the underlying file systems, such as UFS or JFS, and the semantics of the underlying file system are preserved (that is, applications see a UFS file system even though it was mounted as GFS). Converting a file system to a GFS destroys any information about the underlying file system. GFSs can only be mounted cluster-wide, and cannot be mounted on a subset of the systems in the cluster. There must be entries in the /etc/vfstab file on each system in the cluster, and they must be identical. (SunClusters does not provide any checks on this or tools to help manage this.)

Multiported disks can also be part of the GFS, but Sun recommends that only two systems be connected to a multiported disk at a time (see below). Secondary systems are checkpointed by the primary system during normal operation, which causes significant cluster performance degradation and memory overhead. The master system performs all I/O to the cluster file system upon request by the other systems in the cluster, but cache is maintained on all systems that are accessing it.

SunCluster manages the master and secondary systems for multiported disks in a list of systems in the "preferenced" property, with the first system being the master, the next system being considered the secondary, and the rest of the systems being considered spares. If the master system fails, the next system on the "preferenced" list becomes the master system for that file system and the first spare becomes the secondary. This means that the "preferenced" list must be updated whenever systems are added to or removed from the cluster, even during normal operation.

TruCluster offers a cluster file system that 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 64K bytes, are done by the CFS server system upon request by the CFS client systems. Thus, TruCluster generally acts as a proxy file system using network I/O. The only exceptions are applications that have been modified to open the file with O_DIRECTIO. Oracle is the only application vendor that has taken advantage of this.

Veritas offers a cluster file system in two different products. The SANPoint Foundation Suite – High Availability (SPFS – HA) enhances VxFS with cluster file system extensions on HP-UX and Solaris, providing direct I/O to any volume from any system in the cluster that is physically connected to the volume. SPFS requires that any volume managed this way be physically connected to every system in the cluster. This offers direct I/O functionality for the general case of file systems with flat files. For Oracle 9i/10g RAC, the Veritas Database Edition/Advanced Cluster (DBE/AC) for 9i/10g RAC supports direct access I/O to the underlying VxFS. Note that if you do not use either SPFS – HA or DBE/AC, the Veritas Volume Manager defines a volume group as a "cluster disk group" (a special case of a "dynamic disk group"); this is the only type of volume that can be moved from one system in a cluster to another during failover. This is not a cluster file system, since Veritas emphasizes that only one system in a cluster can make use of the cluster disk group at a time.

All of the above systems that implement direct access I/O use a "master" system to perform meta-data operations. Therefore, operations like file creations, deletions, renames, and extensions are performed by one of the systems in the cluster, but all I/O inside the file or raw device can be performed by any of the systems in the cluster using direct access I/O. OpenVMS Cluster Software and PolyServe have multiple "master" systems to optimize throughput and reduce contention.

An advantage of direct access I/O, whether implemented with a file system or with raw devices, is that it allows applications to be executed on any system in the cluster without having to worry about whether the resources they need are available on a specific system. For example, batch jobs can be dynamically load balanced across all of the systems in the cluster, and are more quickly restarted on a surviving system if they were running on a system that becomes unavailable. Notice that the availability of the resources does not address any of the recovery requirements of the application, which must be handled in the application design.

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. There is always a difference in performance and latency between local locks and remote locks (often up to an order of magnitude difference (10x)), which may affect overall performance. You must take this into account during application development and system management.

HACMP offers the Cluster Lock Manager, which provides a separate set of APIs for locking, in addition to the standard set of UNIX System V APIs. All locking is strictly the responsibility of the application. The Cluster Lock Manager is not supported on AIX with the 64-bit kernel. HACMP also offers the General Parallel File System, which was originally written for the High Performance Technical Computing (HPTC) environment but is now available in the commercial space.

MySQL Cluster tracks the locks for the database itself, but does not offer a generalized distributed locking mechanism.

NSK does not even have the concept of a distributed lock manager, as none is required. Ownership of all resources (files, disk volumes, and so forth) is local to a specific CPU, and all communication to any of those resources uses the standard messaging between CPUs and systems. The DAM responsible for a given volume keeps its locks and checkpoints this information to a backup DAM located on a different CPU. Because of the efficiencies of the messaging implementation, this scales superbly.

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.

Oracle implements a distributed lock manager on HACMP, Linux LifeKeeper, SunClusters, and Windows 2000/2003, but takes advantage of the native distributed lock manager on OpenVMS Cluster Software, Serviceguard Extension for OPS/RAC, and TruCluster.

SQL Server 2000/2005 tracks the locks for the database itself, but, because only a single instance of the database can be running at one time, there is no distributed lock manager.

SunCluster and TruCluster extend the standard set of UNIX APIs for file locking in order to work with the cluster file system, resulting in a proxy for, and a layer on top of, the standard file systems. Keep in mind that even though the file system is available to all systems in the cluster, almost all I/O is performed by the master system, even on shared disks.

Veritas uses the Veritas Global Lock Manager (GLM) to coordinate access to the data on the cluster file system.

Windows 2000/2003 does not have a distributed lock manager.

Quorum

When 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, but cannot communicate with each other. This is intended to prevent cluster partitioning, which is known as "split brain."

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" 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 in order to form a cluster. If the actual quorum value is below the required quorum value, 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.

In a running cluster, the number of expected votes is the sum of all of the members with which the connection manager can communicate. As long as the cluster interconnect is working, there are 2 systems available and no quorum disk, so the value is 2. Thus, actual quorum is greater than or equal to required quorum, resulting in a valid cluster.

When 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 the actual number of votes becomes 1 for each of the systems. Applying the equation, actual quorum becomes 1, which is less than the number of required quorum required to form a cluster, so both systems stop and refuse to continue processing. This does not support the goal of high availability; however, it does protect the data, as follows.

Notice what would happen if both of the systems attempted to continue processing on their own. Because there is no communication between the systems, they both try to form a single system cluster, as follows:

1. System A decides to form a cluster, and mounts all of the cluster-wide disks.
2. System B also decides to form a cluster, and also mounts all of the cluster-wide disks. The cluster is now partitioned.
3. As a result, the common disks are mounted on two systems that cannot communicate with each other. This leads to instant disk corruption, as both systems try to create, delete, extend, and write to files without locking or cache coherency.

To avoid this, we use a quorum scheme, which usually involves a quorum device.

Picture the same configuration as before, but now 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. In this case, when the cluster interconnect fails, the following occurs:

1. 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 information, such as the system id of the founding member of the cluster and the time that the cluster was newly formed, to the quorum disk . System A then computes the votes of all of the cluster members (itself and the quorum disk, for a total of 2) and observes that it has sufficient votes to form a cluster. It does so, and then mounts all of the disks on the shared bus.
2. 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 count either system A or the quorum disk's votes in its inventory. So system B then computes the votes of all of the cluster members (itself only

for a total of 1) 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 the cluster-wide disks. If there are other systems in the cluster, the value of required quorum and expected quorum would be higher, but the same algorithms allow those systems that can communicate with the founding member of the cluster to join the cluster, and those systems that cannot communicate with the founding member of the cluster are excluded from the cluster.

This example uses a "quorum disk," but in reality any resource can be used to break the tie and arbitrate which systems get access to a given set of resources. Disks are the most common, frequently using SCSI reservations to arbitrate access to the disks. Server systems can also be used as tie-breakers, a scheme that is useful in geographically distributed clusters.

Cluster Configurations

The following table summarizes important configuration characteristics of cluster products.

	Max Systems In Cluster	**Cluster Interconnect**	**Quorum Device**
HACMP AIX, Linux	32	Network, Serial, Disk bus (SCSI, SSA) (p)	No
LifeKeeper Linux, Windows	16	Network, Serial (p)	Yes (Optional)
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	64	Network	Yes
NonStop Kernel	255	ServerNet (a)	Regroup algorithm
OpenVMS Cluster Software	96	CI, Network, MC, Shared Memory (a)	Yes (Optional)
Oracle 9i/10g RAC Many O/S's	Dependent on the O/S	Dependent on the O/S	n/a
PolyServe Matrix Linux, Windows	16	Network	Yes (membership partitions)
Serviceguard HP-UX, Linux	16	Network, HyperFabric (HP-UX only)	Yes = 2, optional >2
SQL Server 2000/2005 Windows	Dependent on the O/S	Dependent on the O/S	n/a
SunCluster Solaris	8	Scalable Coherent Interface (SCI), 10/100/1000Enet (a)	Yes (Optional), recommended for each multiported disk set
TruCluster Tru64 UNIX	8 generally, 512 w/Alpha SC	100/1000Enet, QSW, Memory Channel (p)	Yes (Optional)
Veritas Cluster Server AIX, HP-UX, Linux, Solaris, Windows	32	Dependent on the O/S	Yes (using Volume Manager)
Windows 2000/2003 DataCenter	4/8	Network (p)	Yes

Figure 4 Cluster Configuration Characteristics

HACMP can have up to 32 systems or dynamic logical partitions (DLPARs, or soft partitions) in a system. Except for special environments like SP2, there is no high speed cluster interconnect, but serial cables and all Ethernet and SNA networks are supported as cluster interconnects. The cluster interconnect is strictly active/passive, and multiple channels cannot be combined for higher throughput. The disk busses (SCSI and SSA) are also supported as emergency interconnects if the network interconnect fails. Quorum is supported only for disk subsystems, not for computer systems.

LifeKeeper supports up to 16 systems in a cluster, connected by either the network or by serial cable. These are configured for failover only, and are therefore active/passive. Any of the systems can take over for any of the other systems. Quorum disks are supported but not required.

MySQL Cluster can have up to 64 systems in the cluster, connected with standard TCP/IP networking. These can be split among any combination of MySQL nodes, storage engine nodes, and management nodes. MySQL uses the management node as an arbitrator to implement the quorum scheme.

NonStop Kernel can have up to 255 systems in the cluster, but, given the way the systems interact, it is more accurate to say that NonStop Kernel can have 255 systems * 16 processors in each system = 4,080 processors in a cluster. Each system in the cluster is independent and maintains its own set of resources, but all systems in the cluster share a namespace for those resources, providing transparent access to those resources across the entire cluster, ignoring the physical location of the resources. This is one of the methods that NSK uses to achieve linear scalability. ServerNet is used as a communications path within each system as well as between relatively nearby S-series systems. ServerNet supports systems up to 15 kilometers and remote disks up to 40 kilometers, with standard networking supporting longer distances. The ServerNet-Fox gateway provides the cluster interconnect to the legacy K-series. The cluster interconnect is active/active. NSK uses a message-passing quorum scheme called Regroup to control access to resources within a system, and does not rely on a quorum disk.

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 (starting in 2005) any combination of Itanium and Alpha systems, in mixed architecture clusters. There are many cluster interconnects, ranging from standard networking, to Computer Interconnect (the first cluster interconnect ever available, which was introduced in 1984), to Memory Channel, and they are all available active/active. The quorum device can be a system, a disk, or a file on a disk, with the restriction that this volume cannot use host-based shadowing.

Oracle 9i/10g RAC uses the underlying operating system functionality for cluster configuration, interconnect, and quorum. The most common type is 100BaseT or 1000BaseT in a private LAN, often with port aggregation to achieve higher speeds. For low latency cluster interconnects, HP offers HyperFabric and Memory Channel, and Sun offers Scalable Cluster Interconnect. Oracle 9i/10g RAC does not use a quorum scheme; instead, it relies on the underlying operating system for this functionality.

PolyServe Matrix Server uses the underlying operating system functionality for cluster configuration and interconnect. PolyServe on both Linux and Windows primarily uses gigabit Ethernet or Infiniband in a private LAN. Matrix Server uses a quorum scheme of membership partitions, which contain the metadata and journaling for all of the file systems in the cluster. There are three membership partitions: all data is replicated to all of them, providing redundancy. One or even two of these partitions could fail, and PolyServe could still rebuild the information from the surviving membership partition. These membership partitions provide an alternate communications path, allowing servers to correctly arbitrate ownership and coordinate the file systems, even if the cluster interconnect fails. It is good practice to place the three membership partitions on three separate devices that are not the devices of the file systems themselves.

Serviceguard can have up to 16 systems, using standard networking or HyperFabric (HP-UX only) as a cluster interconnect, and uses Auto Port Aggregation for a high speed active/active cluster interconnect. A special requirement 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. Serviceguard can use either one or two quorum disks (two in an Extended Campus Cluster), a quorum server that is not a member of the cluster, or an arbitrator system that is a member

of the cluster. The quorum disk, quorum system, or arbitrator 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 system (on HP-UX and Linux), a disk volume (on HP-UX), or a LUN (on Linux). 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. Single quorum servers can service up to 50 separate Serviceguard clusters for either HP-UX or Linux. Note that quorum servers are not members of the clusters that they are protecting.

SQL Server 2000/2005 uses the underlying operating system functionality for cluster configuration, interconnect, and quorum. The maximum number of servers in an SQL Server environment grew from four with Windows 2000 to eight with Windows 2003.

SunCluster can have up to eight systems in a cluster, using standard networking or the Scalable Coherent Interconnect (SCI) as a cluster interconnect. SCI interfaces run at up to 1GByte/second, and up to four can be striped together to achieve higher throughput, to support active/active cluster interconnects. However, Sun only offers up to a four-port SCI switch, so only four systems can be in a single SunPlex domain. Quorum disks are recommended by Sun, and each multiported disk set requires its own quorum disk. So, for example, if there are four systems (A, B, C and D) with two multiported disk arrays (X and Y) where disk array X is connected to systems A and B, and disk array Y is connected to systems C and D, two quorum disks are required.

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 512 systems. This configuration uses the Quadrix Switch (QSW) as an extremely high-speed interconnect. For the commercial market, TruCluster uses either standard networking or Memory Channel as an active/passive cluster interconnect.

Both OpenVMS Cluster Software and TruCluster recommend the use of quorum disks for 2-node clusters, but make it optional for clusters with a larger number of nodes.

Veritas Cluster Server can have up to 32 systems in a cluster of AIX, HP-UX, Linux, Solaris, and Windows 2000/2003 systems. Standard networking from the underlying operating system is used as the cluster interconnect. Veritas has implemented a special Low Latency Transport (LLT) to efficiently use these interconnects without the high overhead of TCP/IP. Veritas implements a standard type of quorum in Volume Manager, using the term "coordinator disk" for quorum devices.

Windows 2000 clusters can have up to four systems in a cluster, while Windows 2003 extends this to eight. Keep in mind that Windows 2000/2003 DataCenter is a services sale, and only Microsoft qualified partners like HP can configure and deliver these clusters. The only other cluster interconnect available is standard LAN networking, which works as active/passive.

Application Support

The following table summarizes application support provided by the cluster products.

	Single-instance (failover mode)	**Multi-instance (cluster-wide)**	**Recovery Methods**
HACMP AIX, Linux	Yes	Yes (using special APIs)	Scripts
LifeKeeper Linux, Windows	Yes	No	Scripts
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	No (MySQL Server)	Yes	Failover
NonStop Kernel	Yes (takeover)	Effectively Yes	Paired Processing
OpenVMS Cluster Software OpenVMS	Yes	Yes	Batch /RESTART
Oracle 9i/10g RAC Many O/S's	No (Oracle DB)	Yes	Transaction recovery

PolyServe Matrix Linux, Windows	Yes	No	Scripts
Serviceguard HP-UX, Linux	Yes	No	Packages and Scripts
SQL Server 2000/2005 Windows	Yes	No	Scripts
SunCluster Solaris	Yes	No	Resource Group Manager
TruCluster Tru64 UNIX	Yes	Yes	Cluster Application Availability
Veritas Cluster Server AIX, HP-UX, Linux, Solaris, Windows	Yes	No	Application Groups
Windows 2000/2003 Cluster Service Windows	Yes	No	Registration, cluster API

Figure 5 Cluster Support for Applications

Single-Instance and Multi-Instance Applications

With respect to clusters, there are two main types of applications: single-instance applications and multi-instance applications. Notice that these are the opposite of multisystem-view and single-system-view. Multisystem-view clusters allow single-instance applications, providing 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 provide failover for high availability, and 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 multiple processes on a single system are running the application, then the application is single-instance.

An example of a single-instance application is telnet. 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 system fails, the users on that system simply log in to the next system in the cluster and restart their sessions. This is simple failover. Many systems, including HACMP, Linux, Serviceguard, SunCluster, and Windows 2000/2003 clusters set up telnet services as single-instance applications 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 by 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 the same set of disks to be offered as services to multiple systems. This requires a cluster file system with a single-system-view cluster, which can be offered either in the operating system software itself (as on OpenVMS Cluster Software) or by other clusterware (as on Oracle 9i/10g RAC). Although HACMP, LifeKeeper, Serviceguard, SunCluster, and Windows 2000/2003 do not support multi-instance applications as part of the base operating system cluster-ware, 3rd party tools can add multi-instance application capability. For example, NonStop Kernel uses messaging and DAMs to provide this functionality.

Applications, whether single-instance or multi-instance, can be dynamically assigned network addresses and names, so that the applications are not bound to any specific system address or name. At any given time, the application's network address is bound to one or more network interface cards (NICs) on the cluster. If the application is a single-instance application running on a single system,

the network packets are simply passed to the application. If the application is a single-instance application running on multiple systems in the cluster (like the telnet example above), or is a multi-instance application, the network packets are routed to the appropriate system in the cluster for that instance of the application, thus achieving load balancing. Future communications between that client and that instance of the application may either continue to be routed in this manner, or may be sent directly to the most appropriate NIC on that server.

Recovery Methods

The recovery method is the way the cluster recovers the applications that were running on a system that has been removed, either deliberately or by a system failure, from the cluster.

HACMP allows applications and the resources that they require (for example, disk volumes) to be placed into resource groups, which are created and deleted by running scripts specified by the application developer. These scripts are the only control that HACMP has over the applications, and IBM stresses that the scripts must take care of all aspects of correctly starting and stopping the application, otherwise recovery of the application may not occur. The resource groups can be concurrent (that is, the application runs on multiple systems of the cluster at once) or non-concurrent (that is, the application runs on a single system in the cluster, but can fail over to another system in the cluster). For each resource group, the system administrator must specify a "node-list" that defines the systems that are able to take over the application in the event of the failure of the system where it is currently running. The node-list specifies the "home node" (the preferred system for this application) and the list of other systems in the cluster for takeover of the application, in order of priority.

For multisystem-view clusters like Linux, recovery is done by scripts that are invoked when the heartbeat messages between the cluster members detects the failure of one of the systems. Two example scripts for Linux are mon and monit.

MySQL Cluster does not support applications as such, since it is focused only on the database. Any applications which connect to the database are external to MySQL Cluster, even if they are running on the same system as the database instance. Failover of connections from application nodes to database nodes is transparent to the external applications, and failover of external application connections to the application nodes is seamless between instances of the database.

For fault-tolerant systems like HP NonStop servers, recovery within a system is done by paired processes, where a backup process is in close synchronization with a primary process, ready to take over in the event of any failure. Data replication is the foundation for recovery between systems.

Oracle does an excellent job of recovering failed transactions. Oracle 9i/10g RAC offers this same type of recovery of failed transactions between systems in a cluster. The simple way to think about it is to understand the actions that a standalone Oracle database would perform if the single system went down unexpectedly and then came back up and began recovery. These actions are the same ones that Oracle 9i/10g RAC performs on the surviving systems in a cluster if one of the systems went down unexpectedly. This was easy for Oracle to implement, as the code was already thoroughly tested, and it is easy for database administrators to understand, as they are already familiar with the way Oracle performs these operations in standalone databases.

PolyServe Matrix HA uses application-specific scripts to perform failover and failback of applications. Failover can be from one named system to another (1:1), from the failed system to a group of systems (1:n), or from a group of systems to another group of systems (n:m).

Serviceguard has extensive tools for grouping applications and the resources needed to run them into up to 150 "packages" that are then managed as single units. A set of attributes are specified for each application package, including zero or more network identities, disk volumes, application services, and other resources required by the application. The system administrator can define procedures for recovering and restarting the applications on one of a prioritized list of systems in a cluster, in the event of server failure, and can also define whether to fail back when the primary node is available, or to operate in "rotating standby" mode.

SQL Server 2000/2005 does not support application failover as such, since it is focused only on the database. Any applications that connect to the database are external to SQL Server, even if they are running on the same system as the database instance, and application failover is handled by the underlying operating system. The connections to the database automatically fail over to the surviving server.

SunClusters has tools for grouping applications into either "failover resource groups" or "scalable resource groups." Failover resource groups perform recovery of single-instance applications running on a single server, while scalable resource groups perform recovery of single-instance applications that are running on multiple servers. Be aware that "scalable" applications are single-instance applications; the tools provide IP redirection, routing, and load balancing only.

Both OpenVMS Cluster and TruCluster multi-instance applications have built-in methods that enable recovery from failing systems. They can monitor some applications and recover them automatically. OpenVMS Cluster Software specifies recovery using the /RESTART qualifier on the batch SUBMIT command; TruCluster does it with the Cluster Application Availability facility.

Veritas has extensive tools for grouping applications and their required resources into "resource groups," and defines "agents" to manage those groups. There are "failover groups," which can run on only one system at a time, and "parallel groups," which can run on multiple systems at a time. Veritas has created agents for many standard applications, including DB2, Oracle, and SAP for the Veritas Cluster Server. Failover of resource groups is extremely flexible, and can be done by priority (the next server on the list is chosen), round robin (the server running the least number of resource groups is chosen), and load-based (where the actual load at the moment of failover imposes predefined limits on the number of resource groups on a system).

With Windows 2000/2003 Cluster Service, there are three main recovery methods:

- Generic application/generic service. This doesn't require development of any kind. There is a one-time registration of the application for protection by Windows 2000/2003. A wizard guides administrators through this process.
- 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 2000/2003 Cluster Service to do application-specific monitoring and failover. Again, this doesn't require any development at integration time, but merely requires registering the application using the custom resource DLL.
- Cluster Service API. The application is modified to explicitly comprehend that it is running in a clustered environment and can perform cluster-related operations (failover, query nodes, and so forth).

Cluster Resilience

One of the major selling points of clusters is high availability. The cluster must be resilient, even when things go wrong, such as system or storage subsystem failures or peak workloads beyond expectations, and even when things go very wrong, such as physical disasters. The following table summarizes the cluster resilience characteristics of cluster products.

	Dynamic Partitions	**Disk High Availability**	**Disk Path High Availability**	**Cluster Network Alias**
HACMP AIX, Linux	DLPARs (AIX 5.2 and later)	RAID-1 (Logical Volume Manager)	Multipath I/O (active/passive)	Not shared
LifeKeeper Linux, Windows	No	Distributed Replicated Block Device (DRBD)	Multipath I/O (active/passive)	Not shared
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	No	Dependent on the O/S	Dependent on the O/S	Not shared
NonStop Kernel	No	RAID-1, Process Pairs	Multipath I/O (passive)	Shared
OpenVMS Cluster Software OpenVMS	Galaxy	RAID-1 (Host Based Volume Shadowing)	Multipath I/O (passive)	Shared

Oracle 9i/10g RAC Many O/S's	No	Dependent on the O/S	Dependent on the O/S	Dependent on the O/S
PolyServe Matrix Linux, Windows	No	Dependent on the O/S	Dependent on the O/S	Dependent on the O/S
Serviceguard HP-UX, Linux	vPars	RAID-1 (MirrorDisk/UX)	Multipath I/O (active)	Not shared
SQL Server 2000/2005 Windows	No	Dependent on the O/S	Dependent on the O/S	Dependent on the O/S
SunCluster Solaris	Hot add/swap	RAID-1 (Solaris Volume Mgr)	Multipath I/O (passive)	Not shared
TruCluster Tru64 UNIX	No	RAID-1 (Logical Storage Manager)	Multipath I/O (active)	Shared
Veritas SPFS HP-UX, Solaris	No	RAID-1 (Veritas Volume Manager)	Multipath I/O (passive)	No (simulated by Traffic Director)
Windows 2000/2003 DataCenter Windows	No	No (RAID-1 NTFS is not supported in a cluster)	Multipath I/O (active/passive)	Not shared

Figure 6 Cluster Resilience Characteristics

Dynamic Partitions

Almost all of the high-end server systems offer hard partitions, which electrically isolate sections of a large SMP system from each other. But some operating environments also have the ability to dynamically move hardware components between these hard partitions without rebooting the running instance of the operating system. These are called soft, or dynamic partitions.

Dynamic partitions protect against peaks and valleys in your workload. Traditionally, you build a system with the CPUs and memory for the worst-case workload, accepting the fact that this extra hardware will be unused most of the time. In addition, with hard partitioning becoming more popular because of system consolidation, each hard partition requires enough CPUs and memory for the worst-case workload. But dynamic partitioning lets you share this extra hardware between partitions of a larger system. 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. AIX 5.2 with DLPARs, HP-UX 11i with vPars, and OpenVMS V7.3-2 with Galaxy offer this functionality, as follows:

	Move CPUs	**Move I/O slots**	**Move memory**	**Share memory between partitions**
DLPARs AIX 5.2	Yes	Yes	Yes	No
Galaxy OpenVMS V7.3-2	Yes	No	No	Yes
vPars HP-UX 11i	Yes	No	No	No

Figure 7 Dynamic Partitioning

Do not confuse dynamic partitions with dynamic reconfiguration. Dynamic reconfiguration refers to the hot-add and hot-swap capabilities of the servers, where CPU boards, memory boards, and PCI boards can be added or removed and replaced without powering down or even rebooting the server. This requires cooperation with the operating systems, but it is not associated with clustering. The GS-

series of AlphaServers, the IBM pSeries, HP Integrity servers, HP NonStop servers, the SunFire 10K, 12K and 15K systems, and the Superdome systems all offer these capabilities, but they have nothing to do with dynamic partitioning. Sun calls this capability "Dynamic System Domains;" HP calls this "instant Capacity" (iCAP); and IBM call this "On Demand."

Both HP-UX and OpenVMS offer tools to dynamically balance CPUs across dynamic partitions. HP-UX Work Load Manager (WLM) works with the Process Resource Manager (PRM) to offer goal- based performance management of applications. CPUs can be moved between vPars in order to achieve balanced performance across the larger system. OpenVMS offers the Galaxy Balancer (GCU$BALANCER) utility, which can load balance CPU demand across multiple Galaxy instances.

Notice that all of the above software runs in the base operating system, not just in the clustering product. DLPARs, Galaxy, and vPars partitions can be clustered just as any other instances of the respective operating systems can be clustered. WLM and the Galaxy Balancer do not require the dynamic partitions to be clustered.

Disk and Disk Path High Availability

The storage subsystem level includes 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. But some of this redundancy requires cooperation from the host operating system, specifically in the area of multipath I/O.

Multipath I/O allows the system to have multiple physical paths from the host to a specific volume, as when multiple host adapters are connected to redundant storage controllers. This is common with FibreChannel, but it is also achievable with SCSI and HP's Computer Interconnect (CI).

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

With passive multipath I/O, only one path is active at one time, but the other path is ready to take over if the first path fails. This is accomplished in the same way as the active multipath I/O, by having the system re-issue the I/O request.

Figure 6 shows whether the operating system supports active or passive multipath I/O. Many operating systems enable multipath I/O using EMC PowerPath, HP SecurePath, or Veritas Foundation Suite. PowerPath, SecurePath, and Veritas File System (using Dynamic Multipathing) allow both passive multipath I/O and static active multipath I/O, where the storage administrator can set the preferred and alternate paths from the host to the storage subsystem for a specific disk volume. For Linux, HP provides multipath I/O through open source FC HBA drivers, and the Linux MD driver for SCSI.

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

The first level of defense against these types of failures is host-based RAID, which performs mirroring or shadowing across multiple storage cabinets.

AIX Logical Volume Manager offers RAID-1 (mirrored disks) with up to 3 copies of any disk, but does not offer multipath I/O. Network Interface Takeover allows the system administrator to configure multiple network adapters, where one or more is designated as a backup for the others (passive multipath), but this is not provided for storage interface devices. HP offers SecurePath on AIX.

HP-UX uses MirrorDisk/UX to maintain up to three copies of the data. The software for enabling active multipath I/O varies depending on the storage system being used by HP-UX. EMC PowerPath is included in the HP-UX kernel to give active multipath I/O to EMC storage arrays; the HP Logical Volume Manager (LVM) gives active multipath I/O to the EVA and XP storage subsystems.

Linux uses Distributed Replicated Block Device (DRBD), where one of the systems writes to a local disk and then sends an update over the network so that the other system can write a copy of that data to its local disk. HP offers SecurePath on Linux.

MySQL Cluster uses the underlying operating system functionality for disk access and does not offer any enhancements in this area. However, MySQL Cluster does allow simultaneous access to the database from multiple systems in the cluster, a scheme that is both highly scalable and highly available.

NonStop Kernel provides data high availability using a combination of RAID-1 (mirrored disks), passive multipath I/O with multiple ServerNet Fabrics, multiple controller paths and so forth, as well as process pair technology for the fault-tolerant Data Access Managers.

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 multipath I/O, with operator-controlled load balancing.

Oracle 9i/10g RAC is entirely dependent on the underlying operating system for all of these features, and does not implement any in the database or clustering software.

PolyServe Matrix Server does not offer RAID functionality, relying instead on the underlying operating system, and contends that dual redundant disks and multiple FibreChannel paths are not required to build high availability clusters. Matrix Server supports various multipath drivers, including PowerPath and SecurePath on Linux and Windows, and QLogic failover driver on Linux.

Solaris Volume Manager offers RAID 1+0, which can maintain up to three copies of any disk, including the system disk. Solaris supports passive multipath I/O with operator-controlled load balancing.

SQL Server 2000/2005 does not offer any capabilities in this area, relying instead on the underlying operating system.

Tru64 UNIX supports RAID-1 and RAID-5 with the Logical Storage Manager, which can protect any disk, including the system root. Up to 32 copies of a disk are supported. However, LSM does not support RAID-5 in a cluster, nor can an LSM volume contain the boot partitions of cluster members. LSM supports active multipath I/O.

Veritas supports RAID-1 and RAID-5 with the Veritas SANPoint Foundation Suite on HP-UX and Solaris. Dynamic Multipathing provides passive multipath I/O.

Windows 2000/2003 supports RAID-1 with NTFS mirroring on standalone systems but not in a cluster. HP offers SecurePath on Windows.

Accessibility of the mirrored volumes by the other systems in the cluster is the same as for any shared volume. AIX Logical Volume Manager, HP-UX MirrorDisk/UX, Linux DRBD, NSK RAID-1, Solaris Volume Manager, Veritas Volume Manager, and Windows 2000/2003 NTFS mirrors do not allow access to the remote copy of the data. OpenVMS Host Based Volume Shadowing and Tru64 UNIX Logical Storage Manager allow access to all systems in the cluster.

Cluster Network Alias

There are three types of cluster network alias:

- The Dynamic Name System (DNS) server has a single network name, which is the alias for the cluster. A list of "A" records specifies the network addresses of the actual systems in the cluster. The DNS server can "round robin" the requests among the list of network addresses, offering simple load balancing. However, in some cases the DNS server has no information about the utilization of a given system in the cluster, even whether the system is running. Requests may be sent to a system that is down at that moment. In other cases, one or more of the systems in the cluster dynamically update the list of "A" records in the DNS server to reflect system availability and load. This offers load balancing and higher availability, as requests will only be sent to systems that are running and able to do the work.
- A network address is bound to one of the network interface cards (NIC) on one of the systems in the cluster. All connections go to that NIC until that system exits the cluster, at which time the network address is bound to another NIC on another system in the cluster. This offers failover only, with no load balancing, but it does allow a client to connect to the cluster without needing to know which NIC on which system is available at that moment.
- A network address is bound to one of the NICs on one of the systems in the cluster. Again, all connections go to that NIC until that system exits the cluster, at which time the network

address is bound to another NIC on another system in the cluster. However, in this case, when a request is received, the redirection software on that system chooses the best system in the cluster to handle the request by taking into account the services offered by each system and the load on the systems at that moment, and then sends the request to the chosen system. This offers high availability and dynamic load balancing, and is more efficient than constantly updating the DNS server.

Transparent failover is important for highly available applications. However a connection-oriented application requires information about the state of the connection between the cluster and the client at the time the NIC or the system failed, which the system must be able to preserve.

Single-instance applications (which can only run on one system in the cluster at a time) have their network address bound to a single NIC, but allow failover of that network address to another NIC on a surviving system. This is indicated by "not shared" in Figure 6. Multi-instance applications can have their network address shared between multiple NICs, which is indicated by "shared" in the table.

MySQL Cluster, Oracle 9i/10g RAC, PolyServe Matrix Server and SQL Server rely on the underlying operating system functionality for network alias, and do not offer any enhancements in this area.

HACMP allows multiple IP addresses to be mapped to the same NIC, which provides cluster alias functionality by using IP Address Takeover (IPAT) to move the network address from a failed system to a surviving system. However, each of these IP addresses must be on their own subnet. (Note that this is the opposite of the Serviceguard requirement.) HACMP does not preserve the state of existing connections during a failover; the client must reconnect to the cluster alias, and the new incoming requests are distributed across the surviving systems.

LifeKeeper allows an additional IP address to be mapped to a NIC, and the failover of that address to a surviving system. LifeKeeper does not preserve the state of existing connections during a failover.

NonStop servers use the NonStop TCP/IPv6 to offer Ethernet failover. Multiple IP addresses can exist on the same Ethernet ServerNet Adapter, and they can be designated as either "non-shared IP" or "shared IP." Shared IP addresses gain scalability because NSK uses all available Ethernet ServerNet Adapters in the system for outbound traffic, which is, effectively, active multipath I/O. NSK does preserve the state of existing connections during a failover.

OpenVMS Cluster Software offers a cluster alias for DECnet (also known as DECnet Phase IV), DECnet-Plus (also known as DECnet Phase V), and TCP/IP. DECnet requires that one or more of the systems in the cluster be a routing node, but allows any or all of the systems in the cluster to accept incoming packets addressed to up to 64 cluster aliases. DECnet-Plus requires that there be an adjacent DECnet Phase V router somewhere on the network, but allows any or all of the systems in the cluster to accept incoming packets addressed to up to 64 cluster aliases. TCP/IP Services for OpenVMS allows the use of DNS load balancing (called DNS Clusters) with either static or dynamic load balancing. TCP/IP Services also offers failSAFE IP, which provides failover for NIC failures. None of these preserves the state of existing connections during a system failover.

Serviceguard allows multiple IP addresses to be mapped to the same NIC as a relocatable IP address. Multiple IP addresses can exist on the same NIC only if they are on the same subnet. (Note that this is the opposite of the HACMP requirement.) Up to 200 relocatable IP addresses can exist in a Serviceguard cluster. In the event of a NIC failure, HP-UX preserves the state of existing connections, but during a system failover, Serviceguard does not. On Linux, Serviceguard implements network redundancy by grouping two or more NICs together in a Linux process known as channel bonding, which can be configured in high availability mode or load balancing mode. In the high availability mode, one interface transmits and receives data while the others are available as backups. If one interface fails, another interface in the bonded group takes over. Load balancing mode allows all interfaces to transmit data in parallel in an active/active arrangement. In this case, high availability is also provided, because the bond still continues to function (with less throughput) if one of the component LANs fails. To achieve highly available network services, HP highly recommends channel bonding in each critical Internet Protocol (IP) subnet in order. Failover from one NIC to another prevents the package from failing over to another node and is transparent to the application.

SunClusters allows multiple IP addresses to be mapped to the same NIC for either "failover resource groups" or "scalable resource groups," discussed in Application Support. IP addresses that belong to failover resource groups are accessed through and run on a single system in the cluster and are

reassigned if that system fails. Scalable resource groups are accessed through and run on multiple systems in the cluster. Solaris does preserve the state of the existing connection in the event of a NIC failure, but SunCluster does not preserve the state of existing connections during a system failover.

TruCluster offers a cluster alias for TCP/IP using either host routing or network routing (only for virtual subnets). Any system in the cluster can register to receive incoming packets, which are then redirected to the correct system in the cluster, which is determined using weighted round-robin scheduling. A service port that is accessed through a cluster alias can be either "in_single" or "in_multi." "In_single" services are accessed through and run a single system in the cluster, which is reassigned if that system fails. "In_multi" services are accessed through and run on multiple systems in the cluster. The maximum number of cluster aliases is controlled by max_aliasid, with a default of 8 and a maximum value of 102,400, although this value may never be reached due to memory constraints. Tru64 UNIX preserves the state of existing connections in the event of a NIC failure by using NetRAIN, but TruCluster does not preserve the state of existing connections during a system failover.

Veritas Cluster Server does not offer a cluster alias as such, but uses a set of front-end systems running the Traffic Director package. This is equivalent to the F5 BigIP or Cisco ReDirector functionality, in that incoming requests are distributed to a large number of servers, using round robin, weighted round robin, least connections, and weighted least connections algorithms. The Traffic Director is designed to work with Cluster Server on any platform, but the Traffic Director itself runs only on Solaris systems.

Windows 2000/2003 Cluster Service allows multiple IP addresses to be mapped to the same NIC as a failover IP address. Windows does not allow the same network address to be assigned to NICs on different servers, and Windows does not preserve the state of existing connections during a system failover.

Disaster Tolerance

Disaster tolerance protects computer operations and data from site-wide disasters. For example, in Florida we worry about hurricanes, especially after the devastation of four major hurricanes in six weeks in 2004. In other areas of the world, we worry about tornadoes or blizzards. Everybody worries about power failures and fires. The only way to protect your valuable data from these types of disasters is to make sure that it is stored somewhere far away, and to keep it up to date as close to real-time as possible. There are many kinds of data replication, but the two major types are physical replication and logical replication.

Physical Replication

Physical replication can be performed either by the operating system or by the storage subsystem.

Some operating systems use the same software that they use for data high availability in the same computer room, except that the second (or, in some cases, the third) copy of the data is in another physical location. Serviceguard uses MirrorDisk/UX, NSK uses host-based replication to create a Nomadic Disk, SunCluster uses Solaris Volume Manager, and OpenVMS uses Host Based Volume Shadowing for OpenVMS. In other cases, the systems use different software than they use for local copies; for example, HACMP uses GeoRM.

Distance limitations are based on storage interconnect physical limits, which for FibreChannel is usually about 100km.

Configuration is the same as with single-room clusters because the connections to the disk systems use standard FibreChannel (with either long-wave GBICs, or FibreChannel over IP (FCIP)). The exceptions to this are HAGEO, which is limited to eight systems in two locations, and SunClusters which is limited to two systems in two locations, with the option of a quorum disk in a third location, separated by no more than 200 kilometers.

By doing the replication over FibreChannel instead of using networking between the servers, you can replicate the information from the local site to the remote site even if some of the systems in the remote site are not working.

Storage subsystems also perform replication, using either HP Continuous Access or the EMC Symmetrix Remote Datacenter Facility (SRDF). Connecting the FibreChannel switches together, exactly as in the operating system replication described above, allows the storage controllers to perform the replication. The advantage of this method is that the host does not have to be aware that

the replication is occurring, which means that any environment can use this, no matter which operating system, or even mix of operating systems, is using the storage system. Another advantage is that there is no load on the computer systems for this replication because it is all being done in the storage controllers. Host-based mirroring requires two or more host I/O operations for every write: one to each volume in the mirror-set. Controller-based mirroring requires only one host I/O operation for every write. The controller takes care of replicating the write operation to all of the local or remote volumes in the mirror-set. The disadvantage is that the target volume of the replication is inaccessible to any host while replication is occurring. To get around this restriction, have the remote site periodically take "snapshots" of the target volume, and then mount the snapshot volume. This gives the remote systems full access to data that is very close to up-to-date, which is sufficient for many purposes.

Failover requires you to explicitly stop the replication process in the surviving data center, and then explicitly mount the storage subsystems on the systems in the surviving data center to get back into production. Failback requires the same type of operation: 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.

Because of the complexity of sharing volumes by multiple systems in a multisystem image cluster, GeoRM, MirrorDisk/UX, Nomadic Disk, and Solaris Volume Manager do not allow access to the remote copy of the data that is mirrored by the operating system; similarly, HP Continuous Access and EMC SRDF do not allow access to the target volumes mirrored by the storage controllers.

Whether it is done by the host operating system or by the storage controller, physical replication offers bidirectional data replication.

Assume you have a production system in Boston MA, which is working fine. You want to safeguard your data, so you decide to build a disaster recovery site in Nashua NH, which is within 100km (60 miles) of Boston.

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

However, you need processing to continue in Nashua even if your Boston data center is unavailable. So you put some systems in Nashua, and physically cable them to the storage. The target volume of the storage-based replication is not available for mounting by the systems, no matter what operating system they are running, so the Nashua site is idle, simply waiting for a signal to take over the operations from Boston. This signal can be automated or manual, but in either case, the storage subsystem would break the FibreChannel 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 Nashua just sitting idle, so you split your workload and give half of it to Boston and half of it to Nashua. Notice that this is a multisystem-view implementation, so the target volume of a physical copy is not available for mounting by the systems. So, just as you duplicated the Boston storage in Nashua, now you duplicate the Nashua storage in Boston, which you then connect to the systems in Boston as well, and you set up replication from Nashua to Boston.

Now you have your Boston production data being replicated to Nashua, and your Nashua production data being replicated to Boston. You could survive the loss of either data center, and have all of your data in the other one.

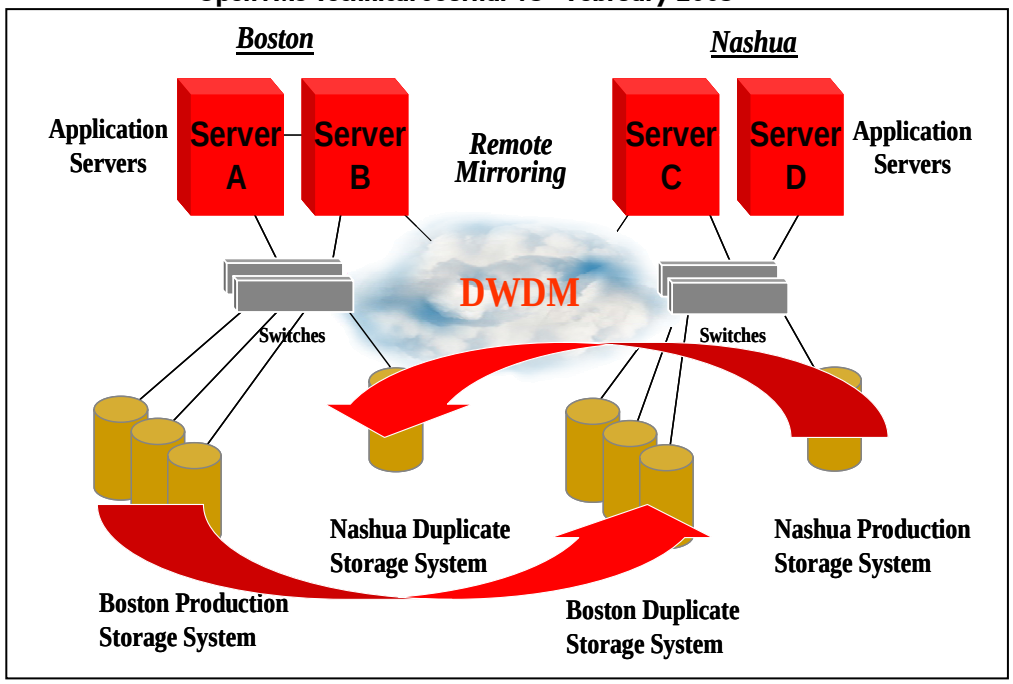

Figure 8 Remote Mirroring

This is known as active/active bidirectional 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 with systems-based data high availability software, or StorageWorks Continuous Access, or EMC SRDF, the data being replicated is actually being written multiple times, by the system or by the storage controller. All of the data on the source disk is being written to the target disk, mission-critical database files as well as temporary files in a scratch area. Careful planning is required in order to ensure that you are replicating everything you need (such as the startup scripts for the database application, which exist outside the database files and are probably not on the same disk volumes as the database files), but are not replicating too much (such as /tmp or paging files).

This is how disaster tolerance works when it is being done by the storage subsystem, or by a multisystem image clustering environment. But some environments have clustered file systems. HAGEO allows raw devices to be accessed on both the local and remote systems, and OpenVMS Host Based Volume Shadowing (HBVS) allows full read/write access to all systems in either site. In these environments, you don't need to have separate storage subsystems that are effectively idle at the remote site; all of your volumes can be in active use. In addition, all of your servers can be used to process all of the information: reads tend to be done by the local disks so they are efficient, and writes are propagated to the other site by the host-based replication technology used in a single computer room.

Another advantage of this solution is that, because OpenVMS HBVS allows three copies of the replicated data, there could be a third site, which would allow continued disaster tolerance for the remaining two sites even if one of the sites was inaccessible.

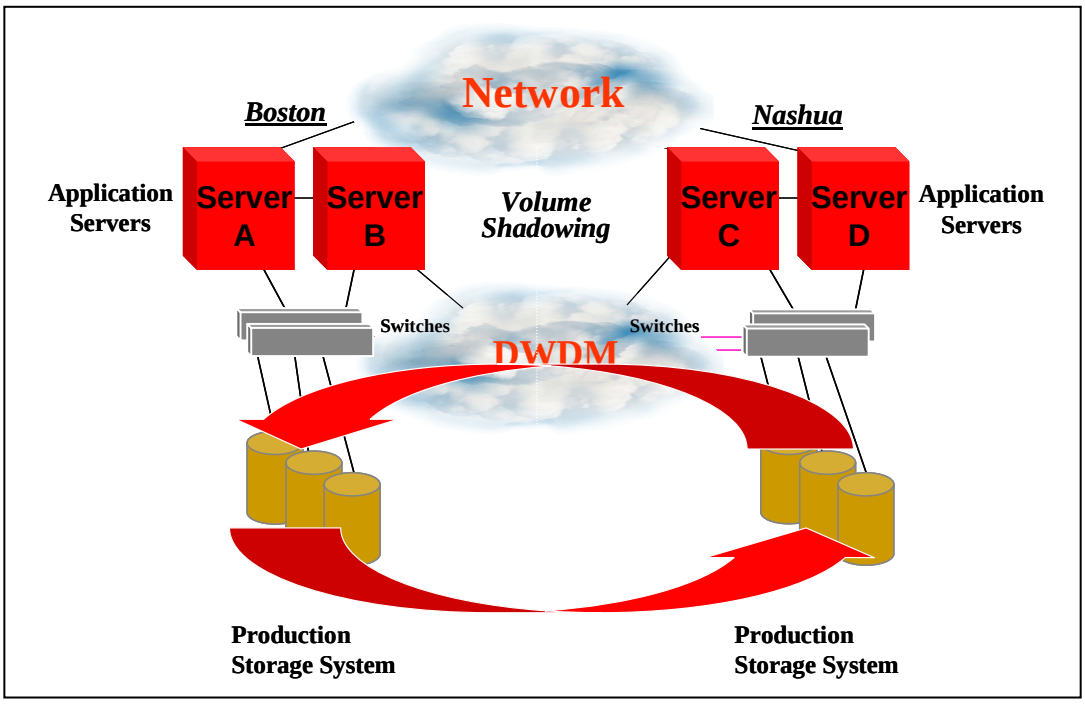

Figure 9 Volume Shadowing

Logical Replication

Logical replication differs from physical replication in both 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.

Recall the example data center in Boston. As before, you need a disaster recovery site. By using logical replication, you can put that disaster recovery site anywhere you want, because you are using standard networking technology. So you choose Los Angeles for your disaster recovery site.

You put a duplicate of your data center in Los Angeles, 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 Los Angeles than in Boston. 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 run some type of production. Notice that it has to be different from the production run in Boston, because the two systems or clusters cannot access the same data. However, you can have bidirectional logical replication, just as you can have bidirectional physical replication. Both data protection and failover capabilities are provided from New York to Los Angeles. This is an active/active logical replication scheme. As in the previous example, failover and failback are semi-automated processes, with some human intervention required.

It is important to know what is being replicated. In the example of physical replication, the storage subsystem takes the bits that were written to the disk and copies them across to the other side. The advantage of this is that everything is being replicated: database files, scripts, flat files, everything. But this approach has some disadvantages as well.

- Everything is being replicated, which might require extremely high bandwidth in order to replicate data that may not needed at the remote site, such as log files.

- A simple operator error (like "rm –r " or "DELETE [...]*.*;*") will be replicated to the target disk in real time.
- Because the storage system is not aware of the database transactions, the information is not guaranteed to be transactionally consistent.

It is impossible to ensure atomicity of a database transaction with physical replication. Physical replication only duplicates the disk blocks that have changed, without regard to the files to which those blocks belong or whether the disk block that was changed was a database row or a database index. For example, a database row may be replicated, but the communications path may fail before the database index information can be replicated, or vice versa. Also, if a transaction spans files or volumes, some parts of the transaction may not be replicated in the proper sequence. If the systems that originated the transaction fail, or if the communications link fails at that moment, the replicated database will probably be corrupted and must be recovered by the database manager before the backup system can begin processing.

Logical replication, however, replicates database transactions, not volumes. It does not replicate security information, patches to the operating system or applications, scripts, or any of the myriad other files needed for maintaining the site; these have to be replicated and maintained by hand or by whole file replication. But the advantage is that an operator error that works on files cannot destroy your entire environment. Furthermore, the logical replication technology is aware of the database transaction as an atomic entity, and so can ensure that the entire transaction is either fully committed or fully rolled back.

Logical replication is done using various technologies. HP Reliable Transaction Router (RTR), HP NonStop Remote Database Facility (RDF) for the NonStop Kernel, IBM MQSeries, and Oracle DataGuard for both Oracle 9i/10g and Oracle Rdb are all examples of logical replication software. Also, many application server suites such as BEA WebLogics, IBM WebSphere, and Microsoft Windows iAS can replicate transactions across multiple sites.

Logical replication has three advantages over physical replication:

- The ability in some cases to read the database at the remote site
- Flexibility of the configurations
- Higher distance capabilities

If the remote database accepts the updates in their original form, as transactions, there is no requirement that the two databases be identical, and because the remote database simply accepts transactions, it can be used at the remote site. In most cases the remote database would be read-only, which is a "best practice" for MQSeries and RTR instead of a requirement, whereas it is a requirement with Oracle DataGuard using logical replication.

If, on the other hand, the remote database accepts updates as transaction logs (also known as "log-shipping" or "log-mining"), the two databases must be identical and the remote database cannot be updated (or even read, depending on the database software) by the remote site. This is the way that RDF and DataGuard work. RDF replicates individual tables, files, or the entire database, and it supports read access to the remote database. Oracle DataGuard using physical replication replicates the entire database, but it does not allow any access to the remote database.

Logical replication can replicate the transaction to multiple sites by duplicating the network packets that carry the update information and sending them to multiple geographically diverse sites. By replicating the transactions, a single database can act as the replication target for multiple other databases. 1:1, 1:n, and n:1 configurations are easily achievable using logical replication technology.

Logical replication usually operates in asynchronous mode, which does not require the remote site to complete the update before continuing the operation at the primary site. Therefore, there is no distance limitation between logical replication sites. However, you should consider the possibility of "in-flight" transactions being lost in the event of a failure of either the primary system or the communications path.

There a distance limitation for physical replication but not for logical replication because physical replication requires a great deal of two-way communication between the sites, and because of the limitation of the speed of light.

In a vacuum, light travels at 186,282 miles per second. Round that off to 200,000 miles per second (to make the math easier), or 200 miles per millisecond. We require confirmation of any message, so we must use round-trip distances. Therefore, light can travel up to 100 miles away and back in 1 millisecond. But light travels somewhat slower in fibre than in a vacuum, and there are the inevitable switch delays, so the conservative rule of thumb is that it takes 1 millisecond for every 50-mile round trip. Therefore, 500 miles adds 10 milliseconds to the latency of a disk access. Given normal disk access latency of 10-15 milliseconds, this merely doubles the latency -- the shadowing software can cope with that. But if the latency is more than that, the software might think the disk at the other end has gone off-line and 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 both the physical and logical asynchronous 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 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 conflict resolution is not necessary. The additional latency of the speed of light is not as important, so the distance limitations are almost nonexistent as long as you don't exceed the distance limitations of your interconnect technology.

If the replication is done synchronously, the write at the local site and the remote sites must be performed before the operation is considered complete. This means that the latency of the transmission is added to the normal disk latency, which slows down each transaction and limits the number of transactions that can be completed in a given period of time. For example, 100km of distance adds slightly over one millisecond to the time needed to complete a write operation, as we discussed before (one millisecond per 50 miles, and 100km is about 62 miles). The advantage of this is that the information is guaranteed to be replicated to the remote site if the operation completed successfully. If the remote site is unable to perform the operation (that is, if a communications failure occurs), the local site can roll back the operation, and the two sites are still synchronized. The disadvantage of this is that each operation will necessarily take longer to perform.

If the replication is done asynchronously, the write at the remote site is queued but is not necessarily performed when the operation is considered complete. The advantage of this is that the local site can continue processing without having to wait the potentially long time until the remote site performs the operation, and eventually the remote site will catch up and be synchronized with the local site. The disadvantage of this is that if the local site becomes inaccessible or physically destroyed before the remote site has received those write operations, they may be lost with no way to recover them. The users at the local site received confirmation that their operation succeeded (and it did, at the local site), but the remote site never got some number of those transactions. This type of problem can be very difficult to resolve.

NonStop Kernel is pioneering a new approach which guarantees no lost transactions while not suffering the latency of synchronous transactions. The transaction logs, also known as audit trails, of the transactions which are being performed at the local site are synchronously replicated at the remote site by using disk mirroring. The primary site can perform all of the transactions at full speed without having to wait for the transactions to be performed by the remote system. If the primary site fails, the remote site can reexecute all of the transactions that took place at the primary site, because it has an up-to-the-last-transaction copy of the transaction log. This technique can be used by any system that has disk mirroring (which may be done by either the host or the storage subsystem) and a transaction log.

Because disk mirroring must be done synchronously in order to guarantee that the transaction log at the remote site has all of the transactions performed by the primary site, the distance is usually limited to about 100 km, but the precise distance varies depending on the replication technology being used.

Disaster Tolerance Features

The following table summarizes the disaster tolerant characteristics of cluster products. Unlike most of the other tables in this paper, this table has multiple options for each operating system, reflecting the multiple technologies that are available as options for each operating system.

Logical replication is done by software outside of the operating system; therefore, all systems are capable of it. Logical replication capabilities are not included in the following chart.

	Data Replication Method/Mode	Link Type and Distance	Cluster Topology and Size
HACMP/GeoRM AIX	Host based physical, sync/async, target is inaccessible	Dark fibre, 103km. Any networking, unlimited distance	Single cluster in 2 sites, 8 systems -or- not clustered
HAGEO AIX	Host based physical, sync/async, target can be read/write	Dark fibre, 103km. Any networking, unlimited distance	Single cluster in 2 sites, 8 systems
LifeKeeper Linux, Windows	Host based physical, sync/async, target is inaccessible	Any networking, unlimited distance	Single cluster in 2 sites, 8 systems
MySQL Cluster AIX, HP-UX, Linux, Solaris, Windows	Host based logical, sync only, target is inaccessible	Any networking, LAN distances	Not clustered
NonStop Kernel, RDF	Host based logical, async only, target is read only	Any networking, unlimited distance	Single cluster in 255 sites, 255 systems
NonStop Kernel, Nomadic Disk	Host based physical, sync only, target is inaccessible	ServerNet, 40km	Single cluster in 2 sites, 255 systems
OpenVMS Cluster Software OpenVMS	Host based physical, sync/async, target is read/write	Dark fibre, 100km. Any networking, 800km	Single cluster in 3 sites, 96 systems
Oracle Data Guard Many O/S's	Host based logical, sync/async, target is read-only	Any networking, unlimited distance	Dependent on the O/S
PolyServe Matrix Linux, Windows	n/a	n/a	Single cluster in 2 sites, 16 systems
Serviceguard Extended Campus Cluster HP-UX	Host based physical, sync only, target is inaccessible	Dark fibre, 100km	Single cluster in 2 sites, 4 systems -or- Single cluster in 3 sites, 16 systems
Serviceguard Metro Cluster HP-UX	StorageWorks Continuous Access, sync -or- EMC SRDF, sync/async	Dark fibre, 100km	Single cluster in 3 sites, 16 systems (3rd site is for arbitrator system only)
Serviceguard Continental Cluster HP-UX	StorageWorks Continuous Access, sync -or- EMC SRDF, sync/async -or- 3rd party host based replication	Dependent on replication technology, effectively unlimited	2 clusters in 2 sites, 16 systems in each site
Serviceguard for Linux with Cluster Extension XP	StorageWorks Continuous Access, sync	Dark fibre, 100km	Single cluster in 3 sites, 16 systems (3rd site is for arbitrator system only)
SQL Server 2000/2005 Windows	Host based logical, sync/async, target is read-only	Any networking, unlimited distance	2 clusters in 2 sites, number of systems is dependent on the O/S
SunCluster Enterprise Continuity Solaris	Host based physical, sync/async, target is read-only	Dark fibre, 200km	Single cluster in 3 sites, 2 systems (3rd site is for quorum

			device only)
TruCluster Tru64 UNIX	Host based physical, sync only, target is read/write	Dark fibre, 6km	Single cluster in 2 sites
Veritas Cluster Server AIX, HP-UX, Linux, Solaris, Windows	Host based physical, sync/async, target is read-only (snapshot)	Dark fibre, 100km. Any networking, unlimited distance	64 clusters in 8 sites, each with 32 systems
Windows 2000/2003 DataCenter Windows	StorageWorks Continuous Access, sync only	Dark fibre, 100km	2 clusters in 2 sites

Figure 10 Disaster Tolerance Characteristics

HP-UX Metro Cluster and Windows 2000/2003 DataCenter support only storage-based physical replication. All of the other products offer host-based storage replication and automated failover/failback, but they vary in how this is implemented.

"Dark fibre" in Figure 10 refers to the FibreChannel links that must be dedicated to this operation, and not shared with any other cluster or other task.

AIX uses host-based Geographical Remote Mirroring (GeoRM) to replicate raw devices, logical volumes, or file systems across 103km using FibreChannel, or across any distance using standard networking. The target of the replication (called the GeoMirror remote site) is not accessible by systems at the remote site. GeoMirror devices can operate in synchronous mode (write the local volume and then write the remote volume, but wait until both writes are complete), "Mirror Write Complete" mode (write the local volume and the remote volume at the same time, but wait until both writes are complete), or asynchronous mode. GeoRM can operate between up to 8 AIX servers, which may but do not have to be in an HAGEO cluster.

HAGEO extends HACMP to a geographically distributed configuration. It uses GeoRM to replicate raw devices, logical volumes, or file systems, but can be set up in a cascading failover or concurrent access configuration using GPFS. This is a single logical cluster in multiple sites, so the same configuration rules for an HACMP cluster apply.

LifeKeeper for Linux and Windows uses SteelEye Disaster Recovery Solution to replicate the file systems across any distance using standard networking. The target of the replication is not accessible to systems at the remote site. This is a single logical cluster that is geographically distributed, so the same configuration rules for LifeKeeper clusters apply.

MySQL Cluster replicates the master database changes to all of the slave databases. These changes are tracked in the "binary log," which is enabled in the master database. The slave databases connect to the master database and are updated whenever a change is made to the master. The master database has no formal connection to the slave databases, and is not aware of how many slave databases exist, so this is not really a cluster. Note that the slave databases must start as perfect copies of the master database. This operates over any distance, although latency might be a factor for the slave databases.

NonStop Kernel uses the NonStop Remote Database Facility (NonStop RDF) software to stream transactions across any distance using standard networking. The target of the logical replication is another database, whose DAMs apply the updates. RDF can replicate the data to multiple sites, and supports 1:1, 1:n, and n:1 configurations. Each system is independent of the others, but they are all part of the loosely-coupled cluster, linked by Expand networking software. This allows the remote system to have read access to the logical standby databases.

In addition to the NonStop RDF for real-time transaction replication, NonStop AutoSYNC software will replicate and synchronize application environments automatically. It monitors user defined files on the source system and uses whole-file replication to update the file on the remote systems. While it can replicate any file type, it is most commonly used for batch files, source files, configuration files, and any flat file that needs to be coordinated between sites.

NonStop Kernel also uses standard host-based disk replication technology across up to 40km of ServerNet to create a Nomadic Disk. The target of the replication is not accessible to systems at the remote site, and it does not have to be in the same cluster. A "zero lost transactions" environment can be created by combining NonStop RDF with Nomadic Disks that contain the transaction logs, to create a physical replication backup to the logical replication.

OpenVMS Cluster Software extends the standard clustering technology to the remote sites, and uses Host Based Volume Shadowing (HBVS) to replicate the disk volumes at the remote sites. The target of the replication is fully accessible by all systems at all sites. This is a single logical cluster that is geographically distributed, so the same configuration rules for OpenVMS Cluster Software apply.

Oracle Data Guard can run with or without 9i/10g RAC, and replicates transactions to other databases at up to nine other sites, as follows:

- Transactions are replicated and copied simultaneously to all of the databases where they are executed directly. The databases are fully accessible at all sites, but conflict resolution may complicate the environment.
- Transactions are replicated to a set of databases that have on-disk structures that are identical to the primary database on a block-for-block basis by applying the journal logs of the primary database to the physical standby databases. These databases are not accessible to systems at the remote sites.
- Transactions are also replicated to a set of databases that contain the same logical information as the primary database but which may have a different set of on-disk structures and are not identical to the original on a block-for-block basis. This is done by "mining" the journal logs and transforming those transactions into SQL statements that are then executed on the logical standby databases. This provides read access to the logical standby databases to systems at the remote sites, so that they can be used for queries and reports while still providing disaster tolerance to the original database.

Note that in any case. only the database information is replicated, not all the other information required for full functionality at the remote site.

PolyServe Matrix Server offers stretch clusters for disaster tolerance by using storage based replication. PolyServe offers no remote replication capabilities.

Serviceguard for HP-UX offers three different approaches to disaster tolerance:

- Extended Campus Cluster uses either MirrorDisk/UX (100km) or Veritas Cluster Volume Manager (10km) to replicate the file systems across FibreChannel. The target of the replication is not accessible at the remote site. This is a single logical cluster, which can have up to four systems at two sites when using a lock (quorum) disk, or up to 16 systems at three sites with the quorum server acting as an arbitrator node. Note that the use of either Veritas Cluster Volume Manager or HP Shared Logical Volume Manager (SLVM) is limited to two systems, one in each site, in an Extended Campus Cluster.
- Metro Cluster uses storage-based replication instead of host-based replication. Whether HP StorageWorks Continuous Access or EMC Symmetrix Remote DataCenter Facility (SRDF) is used, the target of the replication Is not accessible at the remote site. This is a single logical cluster which can have up to 16 systems, but it requires a third site for the quorum servers acting as arbitrator nodes.
- Continental Clusters does not have a native replication technology, but can use any other replication technology such as HP StorageWorks Continuous Access, EMC SRDF, or Oracle DataGuard. The accessibility of the target of the replication, and the distance between sites, is dependent on the technology used for replication. Continental Clusters are two separate clusters, each with up to 16 systems. Unlike the other two methods, which offer automated or manual failover and failback, failover and failback in a Continental Cluster is strictly manual.

Serviceguard for Linux integrates with Cluster Extension XP by using storage based replication instead of host based replication with HP StorageWorks Continuous Access. Similar to Metrocluster, the target of the replication is not accessible at the remote site. This is a single logical cluster that can have up to 16 systems, but it requires a third site for the quorum server acting as an arbitrator node.

SQL Server 2000 has offered replication for many years, either replicating the transactions to a remote database or through log shipping. In both cases the remote database is read-only.

SQL Server 2005 adds database mirroring, which provides "instant standby" with zero data loss and failover times of less than three seconds. This also eliminates much of the scripting involved in transaction replication, requiring only a few SQL statements to set up the mirror, fail over the primary database to the mirror, and fail back the mirror to the primary database. The mirror database is read-only. SQL Server 2005 also adds the concept of the "witness," which is a third instance of SQL Server that acts as a quorum server to prevent split brain. Database mirroring is available in two safety levels: "full" (synchronous replication) and "off" (asynchronous replication). Clients are aware of both the principal and mirror servers, and if the session to the principal server is dropped, the client attempts to reconnect first to the principal and then to the mirror.

Both SQL Server 2000 and SQL Server 2005 are dependent on the underlying operating system for the number of systems in the environment, but each site is a separate cluster.

Sun Infrastructure Solution for Enterprise Continuity uses host-based Remote Mirror to replicate either file systems or raw devices across up to 200km using standard networking technology with the Nortel Networks OPTera 5200. The target of the replication is available at the remote site if it is a raw device, but not if it is a file system, which is the same rule as if the replication were being done inside a single computer room. This is a single logical cluster that is geographically distributed, but there is a further restriction: there can only be a single system at each site. This restriction effectively gives you the choice between high availability (using multiple systems at a single site) or disaster tolerance (using one system at each of two sites). A third site can be used for a quorum device.

TruCluster offers storage-based replication using the StorageWorks Continuous Access to replicate file systems across up to 6km using FibreChannel. The target of the replication is not available at the remote site during the replication, but the source volume can be part of an LSM volume. This is a single logical cluster that is geographically distributed, so all of the configuration rules for a TruCluster still apply.

Veritas offers multiple host-based replication products for multiple platforms:

- Veritas Storage Replicator for Windows offers physical replication across unlimited distances over standard networking. It offers 1:1, 1:n, or n:1 schemes, and operates asynchronously.
- Veritas Volume Manager offers physical replication across FibreChannel using its standard RAID-1 technology for the standard VxFS file system. This is available for AIX, HP-UX, Linux, Solaris, and Windows systems.
- Veritas Volume Replicator is an optional capability of Volume Manager, offering physical replication over standard networking. Volume Replicator can replicate up to 32 remote copies of any volume, either 1:1, 1:n, or n:1. This works either synchronously or asynchronously.

None of the Veritas products allow the target volume to be mounted and used directly by the remote systems, but Veritas offers FlashSnap, which allows the remote system to create a snapshot of the target volume that can then be mounted and used.

Windows 2000/2003 DataCenter does not offer host-based replication across any large distance, but depends on storage-based replication. As such, the target of the replication is not accessible at the remote site during replication. Each site is an independent cluster, and requires manual intervention for failover and failback.

Summary

Every operating system offers a high availability option, but their capabilities vary widely. Some systems offer 2 nodes with manual failover time 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. 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. Still others offer true active/active multi-site clusters over hundreds of miles.

Once the business requirements are determined through a continuity planning program, it is up to you to understand these technologies and select the right solution to satisfy these requirements. But you

must also make sure your senior management are aware of the capabilities and limitations of the technology you have chosen. You can correctly implement a 2-node, manual failover system with no disaster tolerance, but your management may assume it is an infinitely expandable fault tolerant system with unattended zero recovery time, even if the primary datacenter is completely and unexpectedly wiped out. This causes a problem that will only be discovered when the system behaves exactly as you implemented it, and management discovers it does not meet their expectations.

To prevent this problem, document exactly what your chosen solution will and won't do, and get full agreement from management. If they need it to do more, you will have to request more money. In any case, HP can provide the hardware, software, and services to help you design and implement a solution that meets both business and operational requirements.

For more information

On IBM HACMP

- http://www-1.ibm.com/servers/aix/products/ibmsw/high_avail_network/hacmp.html for general information on HACMP
- http://www-1.ibm.com/servers/eserver/pseries/library/hacmp_docs.html for the technical documentation on HACMP
    - Chapter 3, "Types of Cluster Resources" says that only raw devices offer direct access I/O
    - Administration and Troubleshooting Guide, Chapter 9, "Managing Shared LVM Components" says that applications must use the Cluster Lock Manager to arbitrate access to raw devices, and that HACMP does not perform this function
    - Planning and Installation Guide, Chapter 3, "Planning Cluster Networking" describes IP Address Takeover
- http://www-1.ibm.com/servers/eserver/pseries/hardware/whitepapers/dlpar.pdf for a white paper on dynamic LPARs
- http://www.almaden.ibm.com/StorageSystems/file_systems/GPFS/Fast02.pdf for a white paper on the General Parallel File System (GPFS), and http://publib.boulder.ibm.com/clresctr/windows/public/gpfsbooks.html for the documentation on GPFS on the various cluster types. Finally, http://www.redbooks.ibm.com/redbooks/SG246954.html for a specific discussion of implementing Oracle 9i RAC on GPFS
- http://www-1.ibm.com/servers/eserver/pseries/library/hacmp_hiavgeo.html for the technical documentation for HAGEO
- http://www-1.ibm.com/servers/eserver/pseries/library/hacmp_georm.html for the technical documentation for HACMP Geographic Remote Mirror (GeoRM)

On SteelEye 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://h18022.www1.hp.com/solutions/enterprise/highavailability/linux/highperformance/index.html for specifications on LifeKeeper with ProLiant servers
- http://linux-ha.org , specifically "What Linux-HA Can Do Now", and "LAN Mirroring Technologies"
- 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.lifekeeper.com/pdf/literature/se_drs_architecture_overview_final.pdf for information on SteelEye LifeKeeper disaster tolerance capabilities
- http://www.kernel.org/software/mon/ for the Service Monitoring Daemon
- http://www.tildeslash.com/monit/ for information on the Monit Utility
- http://sourceforge.net/projects/ssic-linux for information on the Single System Image Linux project

On MySQL Cluster

- http://www.mysql.com/products/cluster/ for general information on MySQL cluster, along with pointers to white papers and FAQs with more detailed information
- http://dev.mysql.com/doc/mysql/en/Replication.html for information on replication

On HP NonStop Kernel

- http://h20223.www2.hp.com/nonstopcomputing/cache/76385-0-0-0-121.aspx for access to the NonStop Technical Library (NTL) product for full NSK information
- http://h20223.www2.hp.com/nonstopcomputing/cache/76385-0-0-0-121.aspx for details on NSK and high availability
- http://h71028.www7.hp.com/ERC/downloads/RDFSVDS[1].pdf for information on the Remote Database Facility (NonStop RDF)
- http://h20223.www2.hp.com/nonstopcomputing/cache/76385-0-0-0-121.aspx for information on the Parallel Library TCP/IP software for network failover

On HP 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 V7.3-2
- http://h18000.www1.hp.com/info/SP2978/SP2978PF.PDF for OpenVMS Cluster Software V7.3 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
  - http://h71000.www7.hp.com/doc/731FINAL/documentation/pdf/OVMS_731_galaxy_gd.pdf, OpenVMS Alpha Partitioning and Galaxy Guide, Appendix A discusses the Galaxy Balancer program
- For cluster alias, see
  - http://h71000.www7.hp.com/doc/73final/documentation/pdf/DECNET_OVMS_NET_MAN.PDF, section 1.2.5.2 for a description of DECnet Phase IV cluster alias
  - http://h71000.www7.hp.com/doc/73final/6499/6499pro_index.html, section 9.2 and section xxx for a description of DECnet-Plus cluster alias

- http://h71000.www7.hp.com/doc/73final/6526/6526pro.HTML, Chapter 6 for a description of DNS load balancing and TCP/IP Load Broker

- http://h71000.www7.hp.com/openvms/whitepapers/Illuminata.pdf for a white paper written by Illuminata describing the disaster tolerant capabilities of the major clustering products, using OpenVMS Cluster Software as the benchmark

On Oracle 9i/10g Real Application Clusters

- http://otn.oracle.com/products/database/clustering/index.html for general 9i RAC information
- Oracle MetaLink note id # 183408.1 - Aug 2003 discusses the cluster file systems supported by 9i RAC (An Oracle MetaLink subscription is required to access the note)
- http://otn.oracle.com/products/database/clustering/RACWhitepapers.html for a series of technical white papers on 9i RAC
    - http://otn.oracle.com/products/database/clustering/pdf/Oracle9iRAC_BusinessWhitePaper.pdf is an excellent introduction
    - http://otn.oracle.com/products/database/clustering/pdf/rac_building_ha_rel2.pdf discusses the management of 9i RAC and some best practices
    - http://otn.oracle.com/deploy/performance/pdf/FederatedvsClustered.pdf covers a lot of the same topics as this paper, but from a database point of view
- http://otn.oracle.com/deploy/availability/htdocs/odg_overview.html for DataGuard, and defines the difference between physical copy (Redo Apply) and logical copy (SQL Apply)

On PolyServe Matrix Server and Matrix HA

- http://www.polyserve.com/products.html for general information on PolyServe products
- http://www.polyserve.com/products_literature.html to request white papers, case studies and other information on PolyServe products
- http://www.polyserve.com/pdf/mxs_datasheet.pdf, "File System Features" which states that direct I/O (which this white paper calls direct access I/O) as a mount option, and does not require application changes
- http://www.polyserve.com/pdf/matrixha_datasheet.pdf, "HA Features and Benefits" which discusses the replication engine
- http://www.polyserve.com/products_mslinux.html for Matrix Server on Linux
- http://www.polyserve.com/products_mswindows.html for Matrix Server on Windows
- http://www.polyserve.com/products_matrixha.html for Matrix HA

On HP Serviceguard for HP-UX and Linux

- http://www.hp.com/go/ha for general information on Serviceguard and high availability
- http://docs.hp.com/hpux/11i for the complete HP-UX 11i documentation set
- 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/ha/ 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

- Process Resource Manager (PRM) and Work Load Manager (WLM)

- http://www.software.hp.com/cgi-bin/swdepot_parser.cgi/cgi/displayProductInfo.pl?productNumber=B2491BA for MirrorDisk/UX
- http://www.docs.hp.com/hpux/ha/index.html#Quorum%20Server for a discussion of the quorum server
- http://haweb.cup.hp.com/Support/Extended-SG-Clusters/Extended-SG-Clusters.pdf for a discussion of Serviceguard Extension for RAC (SGeRAC) with different volume managers – HP Internal Use Only
- http://docs.hp.com/hpux/onlinedocs/B3936-90065/B3936-90065.html, Managing MC/Serviceguard, "How The Network Manager Works" covers relocatable IP addresses
- http://docs.hp.com/hpux/onlinedocs/T1335-90018/T1335-90018.html, Installing and Managing HP-UX Virtual Partitions
- http://h18006.www1.hp.com/products/sanworks/secure-path/linux.html for SecurePath for Linux

On Sun Microsystems SunClusters 3.1

- http://wwws.sun.com/software/cluster/index.html for information on SunCluster
- http://wwws.sun.com/software/cluster/wp-clustereval/wp-clustereval.pdf for an IDC evaluation of SunCluster, which Sun paid for
- http://docs.sun.com/db/doc/816-3383/6m9lt7uuk?a=view#cacfchja, SunCluster 3.1 Administration and Application Development Guide,
    - "Quorum Configurations" for a description of quorum
    - "Multiported Disk Device Groups" for a description of disk sharing and failover
    - "Multihost Disks" which explicitly says that all I/O is performed by the master system, and that only OPS/RAC allows direct access I/O to cluster file systems on multiported disks
    - "Cluster File Systems" for management of the CFS
    - "Data Services" for shared IP addresses (aka, cluster alias)
- http://docs.sun.com/db/doc/816-3384/6m9lu6fid?a=view#cihcjcae, SunCluster 3.1 System Administration Guide, "How To Add A Cluster File System" for a description of the actions needed on each system in the cluster to create a CFS
- http://www.sun.com/products-n-solutions/hardware/docs/pdf/816-5075-11.pdf, SunFire 15K/12K Dynamic Reconfiguration Utility, Chapter 2, "Introduction to DR on the SunFire 15K/12K", describes Dynamic System Domains and makes clear that this is for dynamic reconfiguration of failed components
- http://www.sun.com/solutions/infrastructure/continuity/index.html for information on SunCluster Enterprise Continuity

On HP TruCluster

- http://h30097.www3.hp.com/ for general information on Tru64 UNIX and TruCluster
- http://h18004.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 and 3.0

- 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://h18004.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
- http://h30097.www3.hp.com/docs/base_doc/DOCUMENTATION/V51B_HTML/ARHGVETE/TITLE.HTM, Chapter 6 for a description of the cluster alias
- http://h30097.www3.hp.com/cluster/tru64_campus_clusters.html for a description of the disaster tolerant capabilities of TruCluster

On Veritas Cluster Server, Database Edition, Veritas Storage Foundation (which was known as the SANPoint Foundation Suite), Storage Replicator, Volume Manager and Global Cluster Manager

- http://www.veritas.com/van/articles/3245.html for a good overview of VCS
- http://www.veritas.com/products/listing/ProductDownloadList.jhtml;vrtsid=ES5UF1DWXTTUPQFIYCLCFEY?productId=clusterserver#whitepapers for white papers on Veritas Cluster Server, specifically the "Cluster Server Technical Overview" discusses the configurations and limits of VCS
- http://ftp.support.veritas.com/pub/support/products/ClusterServer_UNIX/252160.pdf, the Veritas Cluster Server 3.5 User's Guide
- http://eval.veritas.com/downloads/pro//gcm/gcm_wp_tech_overview.pdf for a technical overview of Veritas Global Cluster Manager
- http://eval.veritas.com/downloads/pro/DHBrown_Report.pdf discusses the SANPoint Foundation Suite, and compares it to SunClusters Cluster File System
- http://eval.veritas.com/downloads/pro/sp_fdn_suite_ha/spfs_datasheet_pdf.pdf and http://www.veritas.com/van/products/sanpointfoundationsuite.html for a full description of SPFS - HA
- http://eval.veritas.com/downloads/pro/db_edition/dbed_ac_ds.pdf for an overview of Veritas Database Edition/Advanced Cluster for Oracle 9i RAC. Also, http://eval.veritas.com/downloads/pro/vm-tech_review_guide_050803.pdf describes the differences and restrictions between cluster disk groups and the cluster file system for 9i RAC, and specifically says that Dynamic Multipathing is passive
- http://eval.veritas.com/downloads/pro/vsr/vsr_ds.pdf describes Storage Replicator for Windows
- http://eval.veritas.com/downloads/pro/vcs/vcs_td_datasheet_0802.pdf describes Veritas Traffic Director

On Microsoft SQL Server 2000/2005 Enterprise Edition

- http://www.microsoft.com/sql/techinfo/default.asp for general SQL Server 2000 information, and pointers to white papers and documentation
- http://www.microsoft.com/sql/techinfo/administration/2000/availability.asp for specific high availability features of SQL Server 2000
- http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/failclus.mspx for details on SQL Server 2000 failover clustering
- http://msdn.microsoft.com/library/en-us/replsql/replover_694n.asp for details on SQL Server 2000 replication
- http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/logship1.mspx and http://www.microsoft.com/technet/prodtechnol/sql/2000/maintain/logship2.mspx for a complete

description of log-shipping in SQL Server 2000

On Microsoft Windows 2000/2003 DataCenter

- http://www.microsoft.com/windows2000/en/datacenter for general Windows 2000 DataCenter information
- http://www.microsoft.com/windowsserver2003/technologies/clustering/default.mspx for information on Microsoft clustering options
- http://www.microsoft.com/windowsserver2003/default.mspx for general Windows 2003 information
- http://www.microsoft.com/windowsserver2003/evaluation/features/highlights.mspx#cluster for Windows 2003 Cluster Service information
- http://www.microsoft.com/windows2000/en/datacenter/help/ and http://www.microsoft.com/windowsserver2003/proddoc/default.mspx for the documentation. Specifically:
  - Choosing a Cluster Model, emphasizes that Windows 2000 DataCenter is a multisystem-view cluster, and 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 with Windows 2000, and 8 systems with 2003
  - 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 Server Clusters, How To…, Perform Advanced Administrative Tasks
  - Choosing a RAID Method
  - Disaster Protection discussing the lack of WAN RAID

On HP StorageWorks Continuous Access

- http://h18006.www1.hp.com/storage/software.html for general information on StorageWorks high availability solutions
- http://h18000.www1.hp.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

Acknowledgements

Acknowledgements to Wendy Bartlett, Kirk Bresniker, Dan Cox, Raymond Freppel, Jon Harms, Ron LaPedis, Scott Lerner, Kerry Main, Keith Parris, Bob Sauers, Wesley Sawyer and Chuck Walters for their invaluable input and review.

Revision History

V0.7, 1995 to 2001 – Developed and presented the "Unified Clusters" talk to many groups

V0.8, Sep 2001 – Was scheduled to present "A Survey of Cluster Technologies" at the Encompass 2001 in Anaheim CA, but was interrupted by other events (11-Sep-2001) ☹

V0.9, Oct 2002 – Presented "A Survey of Cluster Technologies" at the HP Enterprise Technology Symposium 2002 in St Louis MO

V1.0, Jul 2003 – Published "A Survey of Cluster Technologies" in the OpenVMS Technical Journal, http://h71000.www7.hp.com/openvms/journal/v2/articles/cluster.html

V1.0, Aug 2003 – Presented "A Survey of Cluster Technologies" at the HP Enterprise Technology Symposium 2003 in Atlanta GA, because the traditional HP customers had not attended HP ETS 2002, and there was a lot of interest in the material

V1.9, Sep 2003 – Added IBM HACMP and Sun Microsystems SunClusters material

V2.0, Nov 2003 – Added Oracle, PolyServe and Veritas material, re-organized the Disaster Tolerance section, presented at the OpenVMS Technical Update Days

V2.1, Feb 2004 – Updated extensively with new information on NSK and Veritas

V2.2, Aug 2004 – Added MySQL Cluster, updated Oracle information for 10g

V2.3 Oct 2004 – Added Microsoft SQL Server 2000/2005

V2.4 Jan 2005 -- Edited for publication in the OpenVMS Technical Journal

OpenVMS Technical Journal V5

Porting the Macro-32 Compiler to OpenVMS I64

Porting the Macro-32 Compiler to OpenVMS I64

John Reagan, Macro-32 Project Leader

Overview

In June 2001, the OpenVMS Engineering organization began porting OpenVMS from the Alpha architecture to the Itanium architecture. One of the key components required was the Macro-32 compiler. Porting the compiler from OpenVMS Alpha to OpenVMS I64 presented several challenges and problems. This paper describes the porting of the Macro-32 compiler from OpenVMS Alpha to OpenVMS I64.

History of the Macro-32 Compiler for OpenVMS Alpha

A large portion of OpenVMS is written in Macro-32. When OpenVMS was first ported from VAX to Alpha, a Macro-32 compiler was created that would accept Macro-32 source code and produce Alpha object files. This enabled much of the Macro-32 source code to be ported from VAX to Alpha with only mechanical changes to the source code. Usually, Macro-32 source code had to be enhanced to include new directives (such as ".call_entry" and ".jsb_entry") and to recode several VAX code constructs that could not be supported by the Macro-32 compiler on Alpha.

Goals of the Macro-32 Compiler for OpenVMS I64

Unlike porting from OpenVMS VAX to OpenVMS Alpha which often required source code changes, our goal for porting Macro-32 code from OpenVMS Alpha to OpenVMS I64 was that modules would simply recompile with no source changes whatsoever. We hoped that the directives that were added to Macro-32 source files when ported from VAX to Alpha would be sufficient for compiling the code on I64. In the end, it turned out that we needed to add a few additional directives and a handful of source modules would need modification to use them.

Organization of the Macro-32 Compiler for OpenVMS Alpha

The Macro-32 compiler has four main phases:

- Phase 1: Source Parser
  The source parser is the same parser that is used by the Macro-32 assembler on OpenVMS VAX. The parser, written in Macro-32, tokenizes the source lines into an intermediate tuple stream. This tuple stream includes instructions, register references, memory references, symbol assignments, conditional compilation information, and labels.
- Phase 2: Flow Analyzer
  The flow analyzer, written in C, analyzes the tuple stream to identify loops, register usage, condition code usage, and breaks up the instruction stream into a sequence of flow blocks where each flow block begins with a label and ends with a branch or call. The register analysis identifies which registers are input, output, read, or written in a flow block. This information is later used to identify registers that can be used by the code generator as short-term scratch registers. The condition code analysis optimizes the generated code by materializing the equivalent VAX condition codes only when they are used by subsequent instructions.
- Phase 3: Code Generator and Register Allocator
  The code generator and register allocator, written in BLISS, processes the flow blocks produced by the flow analyzer. For each VAX instruction in the flow blocks, the code generator produces Alpha instructions to produce the equivalent behavior. The output from the code generator is a list of code cells where each code cell might be an Alpha instruction with its operands or a label. The register allocator keeps tracks of temporary registers used by the code generator. Most of the temporary registers are used only during the code corresponding to a single VAX instruction. However, some of the temporary registers represent condition codes and may have to live for multiple instructions or even around a loop.
- Phase 4: Instruction Scheduler and Peephole Optimizer
  The instruction scheduler and peephole optimizer are provided by the GEM backend code

generator used by the other Alpha compilers. However, in those compilers, the corresponding language front ends produce a sequence of intermediate language tuples and GEM performs its own flow analysis and code generation. In the case of Macro-32, those components of GEM are not used. The Macro-32 compiler produces the list of code cells itself and passes them to the final phases of GEM. The instruction scheduler reorders instructions for better performance. The peephole optimizer removes or modifies related instructions for better performance. GEM also produces the debug information and writes the object file.

What Changed and What Stayed the Same?

For porting the Macro-32 compiler from Alpha to I64, each phase required a different set of changes with two phases requiring almost no work on our part.

The parser required only two source lines to be changed to identify a CALLS instruction to a routine that returned values in registers other than R0 or R1. Such a construct requires the use of a new directive for OpenVMS I64 (the directive is ignored by the compiler on Alpha). The resulting parser is common code between the Alpha and I64 compilers.

The instruction scheduler and peephole optimizer provided by GEM still accepted code cells as input but those code cells now represented Itanium instructions rather than Alpha instructions. There is a different GEM for each target. While the GEM team spent many person years developing the code, from the Macro-32 compiler's point of view, the interface stayed basically the same with the addition of a handful of new flags in the GEM symbol table.

Flow Analyzer Changes

The main tasks of the flow analyzer of grouping tuples into flow blocks and identifying registers used in each block remained essentially the same and required no changes.

However, the flow analyzer also is aware of how many routine arguments are passed in registers and which registers are preserved around routine calls. These values are different from Alpha to I64. For instance, on Alpha the first six arguments are passed in registers while on I64 the first eight arguments are passed in registers. Likewise, on Alpha a standard routine preserves registers R2-R15 by default while on I64 only registers R4-R7 are preserved by a standard routine.

Unfortunately, when the Alpha Macro-32 compiler was written, the information was coded with literal numbers such as "6" or as literal integer or hexadecimal masks. We had to hand examine the entire flow analyzer looking for literals like "5", "6", "7", etc. and change them into symbolic constants which expand to the correct value depending on the target of the compiler.

We added one significant feature to the flow analyzer to deal with a unique feature of the Itanium architecture. Itanium integer registers are actually 65 bits wide. They contain 64 bits of data and another bit called the NaT bit (Not a Thing). This bit identifies a register that has not been initialized and doesn't contain a value. When a register that contains a NaT is written to memory with the regular Itanium store instructions, the hardware raises a NaT Consumption Fault. To avoid this error, there are special Itanium instructions that must be used to spill all 65 bits of a register to memory.

In Macro-32 routines, we found a common practice of what we call a courtesy save. This is code, usually in JSB routines, that explicitly does a PUSHL of some register, then uses that register for some local purpose, and then does a POPL to restore that register. The save sequence is always executed, it does not matter if the caller was actually using that register. On VAX and Alpha, for callers that did not use that register, the save sequence simply saved and restored some random value. However, on Itanium, if that register happened to contain a NaT, the PUSHL would result in a NaT consumption fault. If that occurs in kernel mode, OpenVMS will crash. We found this out the hard way.

We could not use the special 65 bit register save instructions since Macro-32 code that is building a data structure on the stack or is pushing arguments for an upcoming routine call would only expect 32 bits to be written. So we decided that if we could identify registers written to memory that appear to be courtesy saves (writes to memory with no prior writes to that register), then we would generate additional code in the routine prologue to detect a NaT and write a negative one (-1) into that register. In addition, if the register was one of the Itanium preserved registers (R4-R7), we would have to restore the NaT at routine exit.

The flow analyzer was enhanced to find these courtesy saves and propagate the information between flow blocks. The result was a mask of registers that would need NaT Guarding in the routine prologue.

Instruction Generation

The code generator processes each instruction tuple in the flow blocks built by the flow analyzer and generates either Alpha or Itanium instructions. The code generator phase is the only place where we decided not to use common code but have separate modules for Alpha and I64. We cloned the eight Alpha-specific modules and then re-implemented every routine in each module. Some instructions were easy (for example, MOVL and ADDL) while others (DIV, FFS, INSV, EXTZV, EVAX_LDQ_L, and others) took several times to get correct. One subtle difference between the Alpha instructions and the I64 instruction is that literal operands in the Alpha instructions are zero-extended while literal operands in the I64 instructions tend to be sign-extended. We also found several cases where the compiler on Alpha had extended traditional VAX instructions to access part of the Alpha architecture. For example, allowing the BBC instruction to branch on a bit larger than 31 in a register. None of these were ever documented by the Alpha compiler and were only found after problem reports from OpenVMS I64 testing. After that experience, we were more careful about checking our results both against the VAX architecture and the Alpha compiler behavior.

The Alpha compiler has a large set of EVAX_ built-ins to provide access to Alpha instructions. In almost all cases, we support those built-ins by generating one or more Itanium instructions to do the same task. However, some of the EVAX_ built-ins are PAL calls on Alpha. Since there is no PAL code on I64, new system services were added to OpenVMS to provide the similar functionality. The PAL support was removed from the compiler (with the exception of the queue instructions) and new macros were provided in STARLET.MLB that expand to the appropriate system service. The end result is that Macro-32 source code believes that it is using Alpha EVAX_ built-ins to access PAL code when in fact it is calling newly written system services.

Given the relatively small number of available registers on Alpha, the Alpha compiler has extensive code to spill registers to the stack if the generated code requires more temporaries than are currently available. We were able to delete all of that code but had to extend our register allocation mechanisms to include predicate registers as well as distinguishing between the lower 32 static registers and the higher 96 stacked registers. Each routine on I64 can have up to 128 registers. In addition, because the I64 output registers vary based on the most recently executed 'alloc' instruction, we added analysis to ensure that routines that jumped between each other had compatible output registers numbers.

Calling Standard Differences

The Calling Standard for I64 defines registers R8 and R9 as the return value registers. However, all existing Macro-32 source code has assumed that it is dealing with either the VAX or Alpha calling standards and specifies R0 and R1 as the return value registers. Since our goal was to just recompile Macro-32 code from Alpha, we invented a register mapping table. With this register mapping, the compiler on I64 maps all references to R0 and R1 to R8 and R9, respectively. Once we moved R0 and R1, we had to move many of the other registers to make the puzzle all fit together. The end result is that Macro-32 source code believes that it is using Alpha-numbered registers and the compiler's register mapping silently adjusts to match the I64 calling standard. The complete mapping table is available in the HP OpenVMS Macro Compiler Porting and Users Guide in the OpenVMS documentation set.

Most existing Alpha Macro-32 code has been written with the Alpha calling standard in mind. Programmers have assumed that registers R2 through R15 will be preserved by calls to external routines, especially those written in another language like C or FORTRAN. However, on I64, only registers R4 through R7 are preserved by the calling standard. Since we did not want to make people change their source code between Alpha and I64, the compiler automatically preserves R2, R3, and R8 through R15 around each CALLS and CALLG instruction. In general, this behavior is correct. However, for routines that return values in registers other than R0 and R1, the register saves and restores might undo a non-standard return value. In order to support these routines, we had to invent several new directives to provide linkage information about the called routine. These new directives, named ".call_linkage", ".define_linkage", and ".use_linkage", tell the compiler about the non-standard output register usage of the target routine.

In a similar situation, it is possible on Alpha to use a JSB instruction to call a routine that is not written in Macro-32. For instance, if the target routine is written in C, the Alpha calling standard requires the C compiler to preserve registers R2 through R15. Calling this routine with a JSB instruction works correctly. However, on I64, the C compiler preserves only R4 through R7. The Macro-32 program would suddenly find that registers R2, R3, and R8 through R15 would be corrupted by JSB'ing to a C routine. Again, the new directives have an option to indicate that the target routine is written in a language other than Macro-32. In that case, the compiler will preserve R2, R3, and R8 and R15 even around the JSB instruction.

These new directives are also used by the system macros that expand to the system services that replace the Alpha PAL code. For calls to those routines, the compiler must save many more registers to emulate the Alpha behavior of PAL calls not modifying any register, including those higher than R15.

Floating and Packed Decimal Instruction Support

The Alpha compiler supports VAX floating and packed decimal instructions through a set of macros and helper routines written in Macro-64. These routines were re-implemented in Itanium assembly code and the macros were modified to use a different calling sequence.

AMACRO Utility Routines

All of the AMAC$ utility routines from Alpha had to be either rewritten or at least modified to deal with the parameter passing mechanism on I64.

Summary

Porting the Macro-32 compiler from Alpha to I64 was a challenge since we were learning the Itanium architecture and also learning the internals of the Macro-32 compiler. We came close to our original goal of being able to recompile Macro-32 code from Alpha. The vast majority of the Macro-32 code in the OpenVMS source pool recompiled without modification.

Acknowledgements

Peter Haynes and Karl Puder worked on porting the Macro-32 compiler along with the author. Each contributed significant effort to the final compiler. Greg Jordan helped with initial register mapping table. Greg and Christian Moser suggested the Itanium code used to emulate the Alpha load-locked and store-conditional instructions.

OpenVMS Technical Journal V5

Introduction to the Performance Data Collector for OpenVMS (TDC)

Introduction to the Performance Data Collector for OpenVMS (TDC)

Lee Clark, Software Engineer, System Management Tools Group, OpenVMS Engineering

Overview

The Performance Data Collector – **TDC** – Version 2.1 represents an evolution of the TDC V1 project completed during 2001-2002 by OpenVMS Engineering at the request of an independent provider of system management solutions. The approach used for that project was extended and generalized to provide not only a significantly expanded set of system performance metrics but also an open and extensible framework for building performance management and analysis applications for the OpenVMS platform.

TDC provides an Application Programming Interface (API) that can be used to manage collection and processing of approximately1000 performance-related metrics organized into 21 functional categories, or data record types. Among these are storage utilization and I/O; cluster communications; network performance; lock manager performance; process metrics; memory, CPU, server, file system, and cache utilization and performance; paging performance; and SYSGEN parameters. The provided data will not be discussed in this article (see "Software Developers Kit" later in this article).

With its ability to load software modules at runtime, the API supports external development and deployment of additional collection and processing capabilities. The API also supports both file-oriented and "live" processing of collected data.

TDC Version 2.1 is installed as a required System Integrated Product (SIP) with OpenVMS Version 8.2 on Alpha and Integrity Server systems. Software for Alpha systems running OpenVMS Version 7.3-2 is available via download from the web. (The URL is provided at the end of this article.) VAX systems are not supported.

This article begins with an overview of the TDC architecture and programming environment and concludes with a brief discussion of the Performance Data Collector from a system management perspective. Goals of this article are to provide enough information to help you determine whether the software might be useful in your application and to help you determine the level of difficulty you might encounter in integrating the software with your application. This article is not, however, intended as an exhaustive tutorial.

The TDC Software Architecture

The architecture embodies the following key ideas:

- The TDC ***Engine*** manages operations, under the control of a ***Client Application***.
- ***Processor Modules*** produce or utilize data, at the direction of the Engine.
- For any operation, whether formally a data-collection operation or a data-extraction operation, the TDC Engine rigidly enforces a temporal separation between the production of data records and the utilization of those records at each step of the operation.

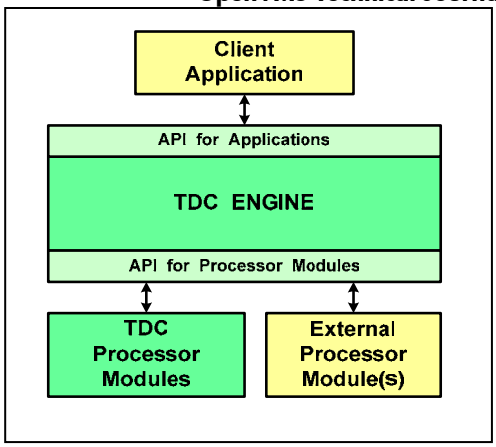

Figure 1 -- TDC Architectural Overview

Figure 1 illustrates the principal components of the architecture and their relationships. (Green boxes in Figure 1 and Figure 2 represent the TDC software; yellow boxes represent software that can be integrated with TDC.) The Client Application depicted in provides a user interface to the data-collection system. The Client might be the control application, TDC$CP, installed with TDC or it might be an independently-developed application—possibly even a server application. Any Client uses the TDC API to control the TDC Engine.

Based on information provided by the Client Application through the API, the Engine (installed as shareable image TDC$APISHR) locates and loads the Processor Modules required to perform the current operation. Processor Modules might be loaded from the library that is provided with TDC (shareable image TDC$LIBSHR), or they might be loaded from one or more externally-developed shareable images specified by the Client Application. From the Engine's perspective, the *only* difference between Processor Modules provided with TDC and "External" Processor Modules is that the Engine knows which shareable image contain those Processor Modules provided with TDC. Once the appropriate Processor Modules have been loaded, the Engine makes no further distinction between TDC-supplied and External Processor Modules.

Processor Modules interact with the TDC Engine through a pointer-based API that overlaps extensively with the name-based API available to Client Applications. Processor Modules respond to messages directed at them by the Engine by performing the specified operation and returning a status to the Engine.

As part of the loading "handshake" between the Engine and a Processor Module, the Processor Module specifies the types of operations it can perform (***capabilities***) and at which points during the current operation it should be called (***synchronization***). Among the possible capabilities that the architecture defines for Processor Modules, the following are the most important in understanding the Engine's organization of the system's runtime behavior:

- A ***Timer*** Processor Module controls the pacing of the current operation. When called by the TDC Engine, a Timer suspends the current operation until it is time to generate a new set of data records. The Timer's suspension of activity might be time-based, it might be based on user input, or it might even be based on interrupts from another application. Although a time-based Timer is provided with TDC, it can be overridden by an externally-provided Timer. Exactly one Timer must participate in data-collection operations. (The Client Application can provide the "timing" function in some modes of operation.) A Timer is optional when extracting data from a file.
- ***Producer*** Processor Modules provide data records to the system. The data records might be collected from the operating system through system services or other means, they might be based on information collected from other applications, or they might be derived from data records provided during the current operation by other Producer Processor Modules. The sole requirement for a Producer is that it provide — through the API — one or more instances of exactly one type of data record whenever the Engine instructs it to do so. The data records

provided by all Producers participating in the current operation are stored in a data aggregate, or *snapshot structure*, by the Engine.

- An ***Extractor*** Processor Module reads data records from a file and uses the API to populate a snapshot structure with them. From the Engine's perspective, an Extractor is nothing more than a "super" Producer: it produces data records of (potentially) several different types when the Engine instructs it to do so. An Extractor is provided with TDC and, at this time, there is no provision to override it with an externally-furnished Extractor.
- ***Consumer*** Processor Modules utilize the data records provided by Producers and made available to them through the API. The Engine maintains two complete snapshot structures: a "current" snapshot containing the most recently-produced set of data records, and a "previous" snapshot containing the set of data records produced prior to that. A Consumer might concern itself either with only the most recent data, or with changes in values from one snapshot to the next. The TDC architecture does not restrict Consumers with respect to how the data is processed. For example, different consumers might write the provided data in raw form into a file, analyze it to create a report, or transmit it to another application.

After all Processor Modules have been successfully loaded, the Engine examines the load-handshake information each Processor Module has provided to determine when, and in which order, the Processor Modules should be called as the current operation proceeds.

The API supports two standard operations: data collections, in which data is collected (by Producers) for processing (by Consumers), and data-extractions, in which data in a file is extracted (by an Extractor) and processed (by Consumers). Although the two operations are distinguished at the API level, the Engine itself operates in much the same manner for the two, as depicted in Figure 2.

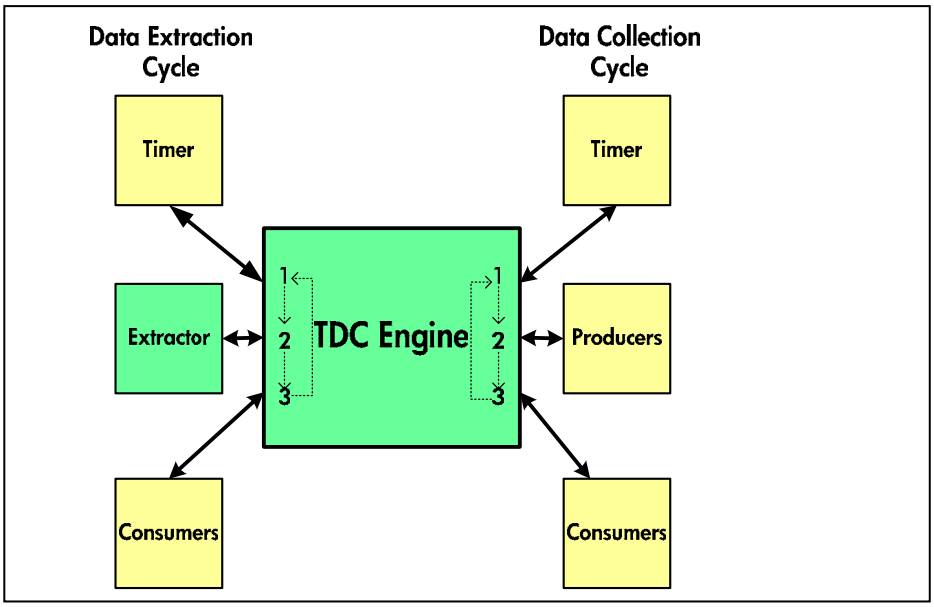

Figure 2 -- Processor Module Calls by the TDC Engine

Figure 2 illustrates how the TDC Engine handles any operation as a repeating cycle of:

1. Calling the Timer (may be absent during extractions) and waiting for it to return control to the Engine.
2. Calling Processor Modules to provide data records (Producers during a collection operation; the Extractor during an extraction operation) to the Engine, which aggregates them into snapshot structures.
3. Calling Consumers to process the data most recently provided by the Producers or Extractor and that has been aggregated into a snapshot structure by the Engine.

A collection operation ends when:

- The specified—usually by the Client Application—number of cycles ("intervals") has been reached, or

- The Timer indicates that the specified—usually by the Client Application—end-time has been reached, or
- The operation is interrupted by the user (Ctrl/Z, Ctrl/C, Ctrl/Y) or by an external source (for example, the TDC control application).

And an extraction operation ends when:

- The specified—usually by the Client Application—number of intervals has been processed, or
- The last interval before a specified—usually by the Client Application—end-time has been processed, or
- All data in the file has been processed, or
- The operation is interrupted by the user (Ctrl/Z, Ctrl/C, Ctrl/Y).

In Figure 2, it is important to note that, in general, any number of Producers can participate in a collection operation, and any number of Consumers can participate in either a collection or an extraction operation. Furthermore, because Consumers only process data residing in snapshot structures provided to them by the Engine, any Consumer Processor Module should be equally at home, whether participating in a collection or in an extraction operation.

Figure 2 contains no indication of how data files are created during a collection operation and read during an extraction operation. Data files are not a part of the overall TDC "architecture." A data file must be created by a Consumer Processor Module, and read by an Extractor Processor Module. The format and content of the data file is then controlled by those Processor Modules rather than by the TDC Engine.

You can develop new software that uses TDC, or integrate TDC into an existing application in either or both of the following ways:

- Write a Client Application to drive the TDC Engine and, by extension, the Processor Modules provided with TDC to collect and manipulate data.
- Develop new Processor Modules to supplement or replace those supplied with TDC. (Note that the TDC Engine and the Processor Modules shipped with TDC are independent entities. In particular, the Engine has no special relationship with, nor dependency upon, any of the Processor Modules shipped with TDC. It could, therefore, be used for a completely different purpose than collecting and manipulating performance data.)

The TDC Programming Environment

A discussion of the TDC programming environment appropriately begins with a description of the environment and the types of software development environments for which it is suited. Key points in this description are the following:

- The TDC API has been developed for use by "C" (and "C++") code. Access to the API by code written with other development languages would require either that "C" wrappers be developed for the API function calls or that the API definition be manually ported to the other language.
- None of the software provided with TDC is multi-threaded, although that need not preclude its use with multi-threaded software. Processor Modules are invoked sequentially at appropriate times, and each usually "has the stage" for the duration of its invocation.
- Processor Modules can, and several of those shipped with TDC do, perform "background" activities to assist in their operations, perhaps to poll for device status or I/O activity. Those TDC-supplied Processor Modules that do background processing use timer events and AST routines for that purpose.
- ASTs are used extensively throughout the TDC software, both for scheduling purposes and for responding to various events. Integrating software that disables ASTs for significant periods of time into the TDC environment is likely to have a deleterious effect on the performance of the TDC software.

- The API is operation-oriented rather than object-oriented. Two operational modes are supported. In one mode, operational parameters are set (by the Client Application), and the Engine then takes control until the operation completes. A step-by-step operational mode, under direct control of a Client Application, is also supported. In general, concurrent operations by a single instance of the software are not supported. Sequential operations are fully supported and are independent of one another because the Engine destroys all context information, including (conceptually) unloading all Processor Modules at the conclusion of each operation.
- Software using the API often has direct access to data structures used by the TDC software. Most operational parameters should not be changed while an operation is under way, but they can be changed freely between successive operations.
- Multiple instances of the TDC software can safely run concurrently within a single OpenVMS system. They do not share data, and, with one exception, one TDC instance will not interfere with another instance's operations. The exception is that the control application shipped with TDC Version 2.1 provides a STOP command that allows one instance of the TDC control application to halt data collection by other TDC control application instances running anywhere in the OpenVMS Cluster.
- The images that comprise the TDC software are not installed as privileged images. Data collection typically requires elevated privileges (refer to "System Management Overview" below), and the required privileges must be enabled by the user prior to running the software. The Engine will not start an operation unless all privileges required by loaded Processor Modules have been enabled (see the "private context" description under "Principal Data Structures" below).

Snapshots and Intervals

The term ***snapshot*** is used in two contexts within the TDC environment:

- The Engine periodically invokes Producer Processor Modules to contribute data records and then invokes Consumers to process those records. A sequence of Producer invocations followed by the Consumer invocations needed to process the collected data is a ***system snapshot***.
- The data aggregate containing all data records produced at a system snapshot is a ***snapshot structure***.

Where context makes the meaning of the term unambiguous, *snapshot* is used in the rest of this article.

An ***interval*** is the period of time that separates successive system snapshots. When data is collected, a "baseline" system snapshot is performed at the start of the operation, and the first interval begins with completion of that baseline snapshot. A TDC data file will, therefore, contain one more snapshot than the number of intervals during which data was collected.

Principal Data Structures

A ***global context*** (type TDC_CTX_t) is the principal vehicle for sharing information among the Client Application, the TDC Engine, and Processor Modules. In general, that context structure is created by the Client Application, which then populates it with parameters appropriate for the current operation, including type of operation (collect or extract), start and end times (for an extraction operation, these specify the time range of the snapshots of interest from the data file), number of snapshots (to collect or to extract), and interval size. There is also provision for status information to be set by the TDC Engine, for operation-progress information to be set by the Engine and by Processor Modules, and for private status and context information to be controlled by the Client Application.

Information in the global context is used by the TDC Engine in preparing to perform an operation, and parts of the global context are used by the Engine to exchange information with Processor Modules as it invokes them:

- The global context provides two pointers to snapshot structures, one for the "current" system snapshot and one for the "previous" system snapshot. Producers typically contribute data records to the current snapshot, while Consumers typically process the data in the current snapshot. The previous snapshot is maintained so that changes in data values can be noted and processed.
- Function pointers stored in the global context by the TDC Engine provide the API through which Processor Modules interact with the Engine (for example, to store or access data records) and with one another (for example, to determine the availability and status of other Processor Modules).
- The global context provides a data buffer into which Producers place data records for storage by the Engine.

An important part of the global context structure is the ***collection header*** (type TDC_CollectionHeader_t). Maintained by one of TDC's Processor Modules (TDC_COLHDR), this area provides basic information about the system configuration and about the state of the operation to be stored as a record in a data file.

Each Processor Module is assigned a ***private context*** area (type TDC_ProcessorCTX_t) used for the exchange of information exclusively between it and the Engine. The private context is used by the Processor Module to declare its capabilities and the synchronization points at which it should be invoked. It is also used to specify any user privileges that the Processor Module requires for the current operation and to specify any other Processor Modules upon which this Processor Module depends. It is through the private context that the Processor Module supplies status information to the Engine. The private context contains a provision for context data that is truly private to the Processor Module as well as for a "public" area for sharing data with other Processor Modules.

A ***processor module description*** (type TDC_ProcessorDesc_t) is created by the Engine as each Processor Module is loaded. It contains relatively static information about the Processor Module (for example, name, shareable image from which it was loaded, version identification, supported data formats).

The data records provided by an individual Producer Processor Module at a single system snapshot are stored as a ***record set*** (type TDC_Set_t). The record set consists of a header that identifies the type of records represented by the set, the times at which the record set was created and last updated, and the count of records currently in the set. The record set header also contains a pointer to an array of pointers to the actual data records that comprise the set.

Record sets can be marked as "*persistent.*" This type of marking is appropriate for relatively static data, such as system configuration information, that is not expected to change frequently during a collection operation.

All record sets contributed by Producers are aggregated into a ***snapshot structure*** (type TDC_SnapshotHeader_t) by the Engine. The snapshot structure consists of a header that contains timestamps that specify when the snapshot structure was created and when it was last updated, as well as a pointer to an array of pointers to record set headers representing the data records that comprise the snapshot.

As it creates a new snapshot structure at the start of a system snapshot, the Engine populates it with any persistent record sets found in the snapshot structure created at the pervious system snapshot. In doing so, the Engine "clones" persistent record sets. A *clone of a record* set receives its own record set header, but receives a copy of the array of data record pointers contained within the record set that has been cloned. The API provides a function to create a clone of a record set; multiple levels of cloning are supported. Note that a clone of a record set is read-only—it cannot be updated with new records.

Processor Modules

A Processor Module is passed three arguments when it is invoked—always in user mode—by the TDC Engine: a pointer to the global context for the current operation, a pointer to the private context created for it by the TDC Engine, and a pointer to the record set it is expected to populate (Producers only). Any Processor Module is likely to be invoked multiple times during any operation. A ***message***

code in the global context (member TDC_CTX_DoWhat) identifies the activity the Processor Module is expected to perform at each invocation. Those message codes include:

- **LOAD** is the first message seen by the Processor Module, at the start of any operation. It is matched by an **UNLOAD** message when the operation has completed. Upon receiving the LOAD message (see Example 2), the Processor Module populates its private context with information that specifies how it will contribute to the current operation (as read from the global context): its capabilities, synchronization points at which it should be called, required user privileges, and the names of any other Processor Modules on which it depends. The Engine uses those names to attempt to locate the specified Processor Modules and also uses them in sorting the full set of loaded Processor Modules into a valid invocation order. (At each system snapshot, a Processor Module is invoked only after invocation of those other Processor Modules on which it depends.) A Consumer also specifies the names of any output formats it supports.
- **INITIALIZE** is the second message seen by a Processor Module for any operation. Upon receiving this message, the Processor Module initializes itself as required, perhaps by allocating memory or other resources. This message is matched by an **END_OPERATION** message at the end of the current operation, at which point the Processor Module frees all resources it might be holding.
- **CONSUME_DATA** is passed once to each Consumer after data has been collected at each system snapshot. It provides the opportunity for the Consumer to process the data just collected (see Example 4).
- **WAITING** is the message passed to a Timer after a system snapshot has been fully processed; it is the signal for the Timer to suspend activity until the next system snapshot occurs.
- Other messages that might be received depend upon the capabilities and synchronization points specified by Processor Modules when they are loaded. Possible *synchronization* points include:
    - *PREBASELINE synchronization*: the Processor Module is invoked prior to collection of the baseline snapshot, with a **BASELINE** message code; this is intended for Timers so that the start of an operation can be deferred.
    - *SNAPSHOT synchronization*: the Processor Module is invoked three times at each system snapshot, receiving the following message codes in the order specified:
        1) **PRESNAPSHOT**: provides an opportunity, if appropriate, to suspend any background activity prior to data collection.
        2) **SNAPSHOT**: Producers provide data records to the Engine for storage (see Example 3). *SNAPSHOT* synchronizers are sent this message only after all such synchronizers have been sent the PRESNAPSHOT message.
        3) **POSTSNAPSHOT**: provides an opportunity, if appropriate, to resume any background activity after all data has been collected. *SNAPSHOT* synchronizers are sent this message only after all such synchronizers have been sent the SNAPSHOT message.
    - *SNAPSHOT_END synchronization*: the Processor Module is invoked after the system snapshot is complete, with a message code of **SNAPSHOT_END**; this synchronization point is used, for example, by the TDC Consumer Processor Module responsible for writing data into a data file.
    - *END_COLLECTION synchronization*: the Processor Module is invoked, with a message code of **END_COLLECTION**, when all system snapshots required for the current operation have been collected and processed; it might be used to close an open data file, or to send notification to another application that the data file is ready for processing.
- A **DEPENDENCY_LOST** message might be received at any time by a Processor Module that has declared itself dependent upon another Processor Module, if that Processor Module has, for any reason, experienced a problem. Data provided with the message serves to identify the Processor Module and the nature of the problem.

- **FILTER** messages are seen by Processor Modules that have declared themselves as ***Filters*** of another Processor Module's data records. Each data record produced by the other Processor Module is passed for examination to the Filter before it is stored in the current snapshot structure by the Engine. The Filter can inspect the record and either allow it to be stored or reject it.

Processor Modules are usually packaged in shareable images. (There is a provision for a Client Application to provide one Processor Module from the client's executable image.) A shareable image can contain multiple Processor Modules. (All Processor Modules shipped with TDC are in a single shareable image.)

The API defines an interface to be used by all Processor Modules. Example 1 illustrates some of the features of that interface.

```
TDC_Status_t XXX_PROCESSOR( TDC_CTX_t *APICtx,       // global context
                            TDC_ProcessorCTX_t *myCtx,//private context
                            TDC_Set_t *mySet, . . .) // record set
```

Example 1 – Interface to a Processor Module

Example 1 shows that a Processor Module receives three arguments (with a provision for optional arguments that is not discussed here) and returns a value of type TDC_Status_t. The first argument points to the global context for the operation; the second points to the Processor Module's private context; and the third usually points to a record set to be populated with data records by the Processor Module. The name for the function through which the Processor Module interacts with the Engine consists of a variable part, "XXX" in the example, followed by "_PROCESSOR." "XXX" is the name by which the Processor Module is known within the TDC system, and the name that a client application uses to specify that the Processor Module be loaded; "XXX_PROCESSOR" is the name by which the TDC Engine attempts to load it from a shareable image.

Example 2 shows how a Processor Module might respond to LOAD messages. In this case, the Processor Module inspects the global context to determine the type of operation to be performed. For a collection operation, the Processor Module indicates, through its private context the following: that it requires CMKRNL privilege, that it produces data records, that it should be called at all SNAPSHOT synchronization points, and that its correct operation depends on inclusion of another Processor Module (in this case, the "PRO" Processor Module supplied with TDC to produce Process metrics). If the Processor Module has dependencies on several Processor Modules, their names are supplied in a comma-separated list (for example, "PRO,DSK" if there were also a dependency on TDC's disk-utilization and performance-capturing Processor Module).

```
    case TDC_K_LOAD:
        if ( APICtx->TDC_CTX_Operation == TDC_K_Collect )
        {
            myCtx->TDC_PCTX_Privileges.prv$v_cmkrnl = 1;
            myCtx->TDC_PCTX_Capabilities = TDC_CAP_M_PRODUCER;
            myCtx->TDC_PCTX_Synchronization = TDC_SYNC_M_SNAPSHOT;
            myCtx->TDC_PCTX_DependsOn = "PRO";
        }
        return TDC_K_STATUS_Success;
```

Example 2 – Handling a LOAD Message

Example 3 illustrates the standard mechanism by which a Producer generates data records and provides them to the TDC Engine for storage. (Error-checking is omitted for brevity.)

```
#include "XXX.h"

case TDC_K_SNAPSHOT:
{
  XXX_Rec *data;
  data = (XXX_Rec *) APICtx->TDC_CTX_DataBuffer;
  for ( int k = 0; k < number-of-records; k++ )
  {
    memset( data, '\0', sizeof *data );
    data->XXX_Prefix.TDC_REC_WU_RecSize = sizeof *data;
    data->XXX_Prefix.TDC_REC_WU_TypeC = XXX_Version;
    data->XXX_Prefix.TDC_REC_A_TypeA =
myCtx->TDC_PCTX_A_ProcessorDesc->TDC_PDESC_A_RecordID.TDC_RecType;
    collect_my_data( data, k );
    APICtx->TDC_CTX_RecordData( APICtx, myCtx, mySet );
  }
}
return TDC_K_STATUS_Success;
```

Example 3 – Storing Data

Each Producer should declare its data record and any other useful information in a *header file* so that the declarations are accessible to Consumers. In this case, the header is "XXX.h", and the data record is declared in that header as type XXX_Rec. At the SNAPSHOT message, a pointer to the XXX_Rec type is declared and made to point to the record buffer supplied by the global context. All data records have a common prefix and, after clearing the data buffer, the record prefix is initialized with a record size, a record-version identification, and a record ID assigned by the TDC Engine that is copied from the processor module description for this Processor Module. A data record is then created and handed off to the Engine for storage through the TDC_CTX_RecordData function pointer in the global context. At that point, any Filters for this record type are invoked by the Engine to accept or reject the record before the Engine actually stores it in the current snapshot.

Example 4 shows how a Consumer might access the data provided by the "XXX" Processor Module shown in Example 3. Here, the Consumer uses the API to locate the record set header for "XXX" data records in the current snapshot. The API call is made through the TDC_CTX_FindInSnapshot pointer in the global context. The Consumer then uses the API to access each record in the record set by means of a series of calls through pointer TDC_CTX_AccessSetRecord in the global context.

Note that the "XXX.h" header file is included when compiling the Consumer shown in Example 4. The declaration of the record type from the header is used to declare an appropriate pointer for use in accessing the data records.

```
case TDC_K_CONSUME_DATA:
{
  XXX_Rec   *data;
  TDC_Set_t *XXX_Set;
  XXX_Set = APICtx->TDC_CTX_FindInSnapshot( APICtx,
                                            APICtx->TDC_CTX_CurrentSnapshot,
                                            "XXX" );
  for ( int k = 0; k < XXX_Set->TDC_SET_LU_RecordCount; k++ )
  {
    data = (XXX_Rec *) APICtx->TDC_CTX_AccessSetRecord( APICtx, XXX_set, k );
    if ( NULL != data )
        processData( data );
  }
}
return TDC_K_STATUS_Success;
```

Example 4 – Accessing Data

Client Applications

Client Applications need not be particularly complex. Example 5 shows a complete Client Application, with error-checking and most infrastructure omitted for brevity. It collects data over a 24-hour period, using separate data files for each hour's data.

```
TDC_CTX_t *APICtx;
char       filename[60];

APICtx = malloc( sizeof *APICtx );
TDC_INIT( APICtx );

for ( int k = 1; k <= 24; k++ )
{
  TDC_REGISTER( APICtx, "DSK", NULL, NULL );
  TDC_REGISTER( APICtx, "XXX", "dev:[dir]myshr", NULL );

  APICtx->TDC_CTX_Operation = TDC_K_Collect;
  APICtx->TDC_CTX_IntervalCount = 30;
  sprintf( filename, "%%N$DAILY-%%D-%.2d", k );
  APICtx->TDC_CTX_DataFile = filename;

  TDC_PREPARE( APICtx );
  TDC_START( APICtx );
  TDC_END( APICtx );
}

TDC_FINISH( APICtx );
free( APICtx );
```

Example 5 – A Client Application

In the example, memory is allocated for the global context, which is then initialized by calling the Engine's initialization function, TDC_INIT().

The Engine is then instructed, by calls to function TDC_REGISTER(), to load TDC-supplied Processor Module "DSK" and externally-provided Processor Module "XXX" from the shareable image at dev:[dir]myshr.exe. Recall from previous examples that "XXX" has a dependency on the "PRO" Processor Module; therefore, the "PRO" module is loaded automatically from the TDC library by the Engine.

Several parameters to define and control the current operation are then specified: it is to be a collection operation that runs for 30 2-minute (the default) intervals (one hour), producing 31 system snapshots. Data collected during the first hour will be stored in file NODE$DAILY-041119-01.TDC$DAT in the current default directory; the second hour's data will be placed in file NODE$DAILY-041119-02.TDC$DAT, and so on. Note that the Engine recognizes three special "escape characters" in file specifications: %N is replaced by the system's node name ("NODE" above); %D is replaced by the current date, or by the start-date specified for the operation, in YYMMDD format (for example, a date of 19-NOV-2004 is placed into the file specification as "041119"); and %Tis replaced by the current time, or by the start time specified for the operation, in HHMMDD format (for example, 15:30:10 is placed into the file specification as "153010").

The Engine is next initialized for the current operation by the call to TDC_PREPARE(), after which the Engine is given control until the operation completes by the call to TDC_START(). A Client Application can retain control over the operation by making a series of calls to TDC_COLLECT_SNAPSHOT() to collect data one snapshot at a time, rather than handing control over to TDC_START().

When the operation completes, TDC_END() is called to free all resources used during the collection. The final status of the operation and other information is available to the Client Application in the global context after TDC_START() returns and before TDC_END() is called.

Because TDC_END() destroys all context information, the Engine must be fully initialized for each operation within the loop; only a single initialization of the global context is required.

When all collections have completed, TDC_FINISH() is called to free any resources still held by the TDC Engine.

Software Developers Kit

The downloadable kit available at the URL specified at the end of this article contains the Software Developers Kit (SDK). The SDK includes header files for all TDC data records (the only documentation for the content and format of those records); a header file with the API's common declarations; complete source code for the TDC control application; sample Client Application code; sample Processor Module code; a "stub" to use as a starting point in developing Processor Modules; miscellaneous support files; and a TDC Programmer's Guide. The Programmer's Guide can also be downloaded separately from the web site.

In planning any development work, consider the information provided by "System Management Overview" below and, in particular, heed the cautions under "Building Software that Uses TDC."

The TDC API

The API is fully documented in the *TDC Programmers Guide* and by comments in the files included with the SDK. Of particular note among the files is TDC_COMMON.H, which declares all the data structures used when interacting with the API. Table 1 below briefly describes the functions that comprise the API.

The names listed under "From Processor Modules" are the names of function pointers in the global context through which Processor Modules make API calls (Processor Modules should never call API functions by name). If there is no entry for a function under "From Processor Modules," then that function is not suitable for use by a Processor Module.

Names listed under "From Client Applications" are the entry names though which Client Applications can make API calls; a Client Application is also free to make API calls through the function pointers listed under "From Processor Modules."

From Client Application	From Processor Modules	Description
Control Functions		
TDC_INIT		Initializes the API.
TDC_REGISTER		Informs the Engine about a Processor Module that should be loaded for the current operation.
TDC_PREPARE		Initializes the Engine for the current operation.
TDC_START		Performs the current operation.
TDC_COLLECT_SNAPSHOT		Performs one step of a collection operation.
TDC_READ_SNAPSHOT		Performs one step of an extraction operation.
TDC_CONTINUE		Transitions from step-by-step processing mode to surrendering control to the TDC Engine to continue and complete the current operation.
TDC_END		Completes the current operation.
TDC_FINISH		Completes use of the API.
TDC_HALT_OPERATION	TDC_CTX_HaltOperation	Requests that the current operation halt (an unscheduled halt).

From Client Application	From Processor Modules	Description
TDC_TEST_COLLECTION_STATUS	TDC_CTX_TestCollectionStatus	Tests whether the current collection operation has been requested to halt.
Snapshot Manipulation Functions		
TDC_NEW_SNAPSHOT		Creates a new snapshot structure
TDC_RESERVE_SNAPSHOT	TDC_CTX_ReserveSnapshot	Prevents freeing of a snapshot structure, to permit continued access to it.
TDC_RELEASE_SNAPSHOT	TDC_CTX_ReleaseSnapshot	Allows freeing of a snapshot structure previously "reserved."
TDC_FIND_IN_SNAPSHOT	TDC_CTX_FindInSnapshot	Locates the record set for a data type within a snapshot structure.
Record Set Manipulation Functions		
TDC_ACCESS_RECORD_SET	TDC_CTX_AccessRecordSet	Locates the "k^{th}" record set within a snapshot structure.
TDC_ACCESS_SET_RECORD	TDC_CTX_AccessSetRecord	Locates the "k^{th}" record within a record set.
TDC_NEW_RECORD_SET	TDC_CTX_NewRecordSet	Creates a new record set.
TDC_CLONE_RECORD_SET	TDC_CTX_CloneRecordSet	Creates a "clone" of a record set.
TDC_NEW_RECORD_SET_IN_SNAPSHOT	TDC_CTX_NewRecordSetInSnapshot	Replaces one record set in a snapshot with another.
TDC_FREE_RECORD_SET	TDC_CTX_FreeRecordSet	Frees memory used by a record set.
Processor Module Manipulation Functions		
TDC_GET_RECORD_TYPE	TDC_CTX_GetRecordType	Gets the ID number assigned to a record type for the current operation.
TDC_GET_RECORD_NAME	TDC_CTX_GetRecordName	Gets the name of the record type associated with an ID number.
TDC_CALL_PROC_MODULE	TDC_CTX_CallProcModule	Allows indirect calls to a specific processor module.
TDC_GET_INFO	TDC_CTX_GetInfo	Returns information about the Engine state (38 items) or about Processor Modules (37 items).
TDC_SET_ATTRIBUTES	TDC_CTX_SetAttributes	Allows modification of some operational parameters of a Processor Module (11 items)
Data Record Manipulation Function		
	TDC_CTX_RecordData	Gives a data record to the Engine for storage.
File-Oriented Functions		
	TDC_CTX_OpenCollection	Opens a collection file.
	TDC_CTX_CloseCollection	Closes a collection file.
	TDC_CTX_WriteCollectionRecord	Writes a data record into a collection file.
	TDC_CTX_ReadCollectionRecord	Reads a data record from a collection

From Client Application	From Processor Modules	Description
		file.
TDC_CREATE_FILE_NAME	TDC_CTX_CreateFileName	Creates a file spec, including substitutions for %N, %D, and %T.
Utility Functions		
TDC_GET_TIME_DIFF_SECS	TDC_CTX_GetTimeDiffSecs	Calculates the difference in seconds between two timestamp values.
	TDC_CTX_DisplayTime	Returns a text string that represents a timestamp value.
	TDC_CTX_TraceLog	Produces tracing and logging messages.
	TDC_CTX_WriteFormatted	Writes formatted output to a file.

Table 1 – TDC API Functions

System Managment Overview

TDC V1.0

You might have installed TDC Version 1.0 for use with a third-party system management solution. TDC Version 2.1 is not compatible with software that makes use of TDC Version 1.0. However, installation of TDC Version 2.1 does not remove files installed by TDC Version 1.0, nor does it impact your ability to continue using the TDC Version 1.0 software with your third-party system management solution. If you rely on software that makes use of TDC Version 1.0, do not remove TDC Version 1.0 from your system until your third-party software has been updated to support use of TDC Version 2.1.

Version 2.1 Files

The Performance Data Collector consists of a control application, an execlet, two shareable images, and various support files. To enable installation in various types of shared-system environments, image file names reflect both the platform for which each is intended and a TDC build identification. Thus, TDC$APISHR$I_V820-0070.EXE is an image for Integrity Server (I64) systems ("$I") running OpenVMS Version 8.2 ("_V820") and provided by the installation of TDC Version 2.1-70 ("-0070").

Version 2.1 Kit Variants

For logistical reasons, two distribution variants of the TDC software exist: **TDC_RT** for the software installed with OpenVMS, and **TDC** for the software downloaded from the web site (URL at the end of this article). TDC_RT contains the software required to run TDC on a specific platform, while TDC contains software to run TDC on any supported platform. Assuming a common TDC build identification (*for example,* Version 2.1-70), there is no functional difference between the software installed by TDC_RT and the software that is installed on the same system by TDC.

The downloadable TDC kit will be revised as needed to deliver maintenance updates and functionality enhancements, while the TDC_RT variant will be updated only with new OpenVMS releases. Installation of an updated TDC variant might or might not remove files previously installed by TDC_RT, depending on your system configuration. Since both variants utilize the same startup file (TDC$STARTUP.COM), which initializes the system environment to use the most recent TDC images available on the system, an older TDC_RT installation and a newer TDC installation can safely co-exist. Do not remove TDC_RT from your system, even if you have installed an updated TDC variant.

Installing TDC in an OpenVMS Cluster

By default, both TDC_RT and TDC install files into SYS$COMMON:[TDC]. Your cluster environment might be set up such that SYS$SYSROOT applies first to a node-specific root (for example,

SYS$SPECIFIC:), then to a shared root (for example, SYS$COMMON), and finally to a clusterwide root (for example, CLU$COMMON). The downloadable TDC kit can be safely installed into the most widely-shared root (for example, CLU$COMMON) that is convenient, provided that installations of the downloaded TDC kit are always performed from the same cluster node so that the OpenVMS product database is consistent.

Building Software that Uses TDC

For the reasons cited above, an individual system might have multiple versions of the TDC software installed. That should not present a problem if you follow a few guidelines in developing and deploying your software:

- Make sure that the SYS$STARTUP:TDC$STARTUP.COM has been run before you build or run your application.
- Always access the TDC software through one of the logical names defined by TDC$STARTUP.COM (TDC$APISHR for the Engine; TDC$LIBSHR for the library of Processor Modules), rather than by specific file names. This will prevent possible problems with multiple simultaneous activations of different TDC images within your application.
- For the same reason, do NOT link shareable images that contain your own Processor Modules against the TDC Engine (TDC$APISHR).
- Link Client Applications against the TDC Engine (TDC$APISHR), but, in general, not against shareable images containing Processor Modules.

Running the TDC Control Application

You can use the Performance Data Collector control application, TDC$CP, to process data files and for various administrative tasks, as well as to collect data. Unless you are running the TDC control application, TDC is not actually "running" on a system. (In other words, it does not run in the background just because OpenVMS is running).

Data collection and some administrative tasks require you to enable various privileges before running the application. The full set of privileges required to perform all tasks and collect all data includes CMKRNL, LOG_IO, NETMBX, PHY_IO, SYSLCK, SYSPRV, and WORLD; SYSNAM privilege is required to run the startup procedure, SYS$STARTUP:TDC$STARTUP.COM. The control application is not installed with the required privileges; HP strongly recommends that you do not install it as a privileged image.

Before starting the control application, each user must define the "TDC" command as shown in Example 6.

```
$ @SYS$STARTUP:TDC$STARTUP ! if not done at system boot
$ SET COMMAND SYS$COMMON:[TDC]TDC$DCL
```

Example 6 – Defining the TDC command

The user can then either run the control application interactively, by typing just TDC at the DCL prompt, or extend the TDC DCL command with a TDC command and qualifiers to specify a complete operation. In the latter case, control is returned to DCL after the operation completes. In interactive mode, you can enter a series of TDC commands before exiting the control application.

Once running, the control application can run collection operations in detached processes, on the local node, on specified nodes within the cluster, or on all nodes within the cluster. The control application can also be used to identify running collections, obtain status from running collections, or stop running collections anywhere within the OpenVMS Cluster.

For more information

For general TDC support issues, use standard OpenVMS support channels.

For software-development questions and issues (only), send email to VMSTDCV2@hp.com.

Software updates and documentation are available for free download at:
http://h71000.www7.hp.com/openvms/products/tdc/

Note: The software kit available at the web site contains runtime software for all supported OpenVMS platforms as well as the Software Developers Kit (SDK). The software kit is packaged one way for downloading and unpacking on Alpha platforms, and a different way for downloading and unpacking on Integrity Server platforms. The contents of the two packages are identical; in particular, both packages provide TDC software for both Alpha and Integrity Server environments. The PCSI-based installation procedure allows you to select the runtime environments you want to install and whether or not to install the SDK.

OpenVMS Technical Journal V5

Are you Certifiable?

Are you Certifiable?

John Gillings

Overview

Certification has become a normal part of the IT industry. It provides a means for people to validate their skills and competencies, and for employers to gain some assurance that their employees really know what they're doing. Typically gaining a certification involves passing an exam that tests a set of competencies in a particular product or technology. In OpenVMS, there are three levels of certification:

- Certified Systems Administrator – CSA OpenVMS v7
- Certified Systems Engineer – CSE OpenVMS v7
- Accredited Systems Engineer – ASE AlphaServer + OpenVMS v7

The underlying exams for these certifications are designed to test experience in OpenVMS (rather than an ability to "cram").

This paper discusses the process used to create the exams and to show why the exams are valid and worthwhile. It should also give you some insight into what to expect when taking a certification exam and provide some tips on exam preparation and technique.

Exams

Although not ideal for all purposes, multiple-choice exams are the only practical and cost-effective means for en-masse testing. They have the advantage of objectivity, repeatability and automation of grading. The same exam may be given anywhere on the planet without concern for biases. They are, therefore, the standard means for examining certification candidates.

In the past, the creation of multiple-choice exams was fairly hit or miss. Just make up a bunch of questions and hope they test what you want to test. Today, it's done using a highly structured methodology, with feedback and statistical validation. A multidisciplinary team consisting of statisticians, psychometricians (who deal with generic issues surrounding exams), and Subject Matter Experts (SMEs) create the exams. In the case of OpenVMS exams, SMEs were drawn from both HP internal and external sources. Specialists were from Services, Engineering, Education, Presales and external Partners.

The Process

Stating the obvious - the most important aspect of creating an exam is to know exactly what it is you want to examine. The first step in the process is to produce a "Competency Model", which is a list of competencies and skills that you expect the candidate to have mastered. This model is tree structured, starting with general areas and branching down to specifics.

Once the competency model is complete, the next step is to weight the branches according to importance. For example, two branches in an OpenVMS competency model might be "Queuing" and "Security". Security might be deemed more important and, therefore, given a higher weighting. This leads to an **exam blueprint**, which is essentially the competency model with percentages attached to each branch and distributed down to the leaves. The blueprint also determines the size of the exam – that is the duration and number of questions with which the candidate is presented.

From the exam blueprint, the SMEs can start writing questions (known as **items** in exam jargon). Each item addresses a specific competency objective from the blueprint, and the number created for each competency branch is determined from the weightings in the blueprint. The typical target is for two versions of the exam (or **forms** in exam jargon) so that a candidate taking the exam a second time receives a different set of questions. Because many items are expected to be "lost" (for a variety of reasons, covered below), it's usually necessary to create at least three times the final number of items required for one form. Items must be distributed according to the blue print weightings, but note that in the final exam, there may not be enough questions to cover all competencies. The exam is really just a sample, rather than a comprehensive coverage of all competencies.

Once the item pool is complete, it is offered for beta test. Volunteer candidates are invited to take the exam to see how it performs. There are three parts to the beta test – first a demographic survey (to determine the candidates' self-assessed level of skill in the product); second, answering all items in the complete item pool; and third, comments and feedback.

Analyzing Beta Test Results

At the completion of the beta test, the results are analyzed. Candidates are divided into groups according to their results and demographics. We expect those who report more experience in the product to receive better marks than those with little or no experience.

Individual items are examined statistically to see how they performed across the beta group. The simplest statistical measure is called **p**, the percentage of candidates who answered the item correctly.

The second measure is called the **point-biserial**. It's a kind of a correlation co-efficient. Items that tend to be answered correctly by the high-scoring candidates and incorrectly by low-scoring candidates have higher positive values. These items discriminate "good" from "bad" candidates. Those with a flat distribution don't discriminate between the groups and have low or zero point-biserial values. Sometimes an item may have a negative value, indicating that the low-scoring candidates answered correctly while the high scorers did not.

The third measure is called **r**. It measures a confidence interval for the distribution of the item answers.

Items that fall outside threshold values for these three measures are dropped from the pool. For example, Items with $p < 25\%$ are considered "too hard" (or perhaps the expected answer is incorrect) and those with $p > 90\%$ are considered "too easy". Similarly, those items with low or negative point-biserial, or low r, are rejected. Any remaining items, over the target requirements are then considered. Better performing items are retained, subject to maintaining the target weightings from the exam blueprint.

For the OpenVMS exams, most items passed the validity tests, so the rejection rate was surprisingly low. Therefore, numerous items, which were well within acceptable performance levels, had to be rejected, because they were surplus to the requirements. Many are included in the Exam Preparation Guides as sample/practice items. EPGs are available from the HP certification web site: http://www.hp.com/certification/

Once the final item pool has been selected, statisticians distribute items among the exam forms, so that they can show statistically that a given candidate is expected to obtain the same mark for each exam form.

The exam team then examines the results, looking in particular at the candidates in the middle of the sample. Using both the results and the demographics of the candidates, a pass mark is selected.

The exam is then released.

Example 1 provides an item that was part of the beta test.

Which LAT feature is used by the cluster alias?

A. auto connection failover

B. auto circuit failover

C. dynamic load balancing

D. DECnet cluster database

Rsp	N	P-Val	P-Bis	Av Time	0-42	43-51	52-62	63-68	69+
A	12	0.13	-0.196	42.8	4	3	3	2	0
B	10	0.11	-0.085	65.8	2	4	2	1	1
C<	61	0.68	0.420	38.6	6	12	12	14	17
D	7	0.08	-0.383	23.0	6	0	1	0	0
Totals		90		0.420					

Example 1 – Sample item with Beta Statistics

Example 1 is a real exam item with beta test statistics. The correct option (C) has a positive point biserial, and all incorrect choices have negative point biserial. P=.68 for the correct answer makes this a reasonably easy question, but it clearly discriminates the strong from the weak candidates. This item was dropped from the final exam pool because there were better performing items available to cover this competency objective.

Writing an item

More jargon! An item consists of the question (known as the **stem**), and some number of potential answers (choices), one of which is correct and the others are distracters. In practice, generating the stem and the correct answer is fairly easy. Finding good distracters is the hard part.

The science of psychometrics gives us some clues to help generate good items. Here are some examples of rejected items that (negatively) demonstrate some of the factors in creating good exam items.

The AUTHORIZE utility can be used to maintain:

A. The password file

B. The User Authorization File only

C. The User Authorization File, the Rights list Database, and the network Proxy database

D. User directories

Example 2 – Length rule

What's wrong?

Although Example 2 is a perfectly valid question, and germane to OpenVMS management skills, the psychometricians will tell us that just about ANYONE, regardless of their OpenVMS skills, would have guessed "C" as the correct answer, simply because it's substantially longer than all the other options. In general, avoid anything that makes the right answer look different from the other choices.

When do you usually need to register a Product Authorization Key (PAK)?

A. After you install the product software

B. Before you install the product software

C. Before you read the product documentation

D. After users begin to use the product

Example 3 – One right, all the rest wrong

What's wrong?

In Example 3, the answer is somewhat subjective. The expected correct answer is choice B (which is arguably the "most correct"), but none of the other choices, (even D) are definitely wrong. To be valid, items must have only one objectively correct answer.

Which utility converts NCP commands to the equivalent NCL commands?

A. DECmigrate

B. DECnet_Migrate

C. NCL_Migrate

D. NCP_Migrate

Example 4 – Trivia quiz

What's wrong?

Although the correct answer for Example 4 is B, this is really a trivia question. If you don't "just know" the answer, there's no way to derive the correct response using your knowledge and experience. Questions like this tend to perform poorly. Because we're not trying to examine rote memory, this type of question does not contribute to the objectives of the exam.

Guidelines

Some other psychometric guidelines include:

- Avoiding negatives in the stem, for example, "Which command is **not**…"
- Avoiding culture specific terminology (slang words**,** jargon**,** references to holidays, Latin abbreviations i.e.; via; sic; status quo; bona fide; et al, culture specific geography)
- Avoiding acronyms and abbreviations

Other item formats that have proven to perform poorly are:

- True/false questions
- Choices "all of the above" and "none of the above"
- Any choice that includes references to other choices like "A and C"

The item writing team includes a psychometrician who ensures that all items conform to the writing standards. The team includes members from around the world, which helps ensure that items are as culture neutral as possible (also note that the beta results include analysis to detect any culture bias in the results).

Taking Exams

Before taking an exam, make sure you've read the Exam Preparation Guide from http://www.hp.com/certification/. The EPG contains references to many manuals. When an item is written, it's necessary to provide a reference to validate the correct answer. All referenced manuals are included in the EPG for the exam. Be aware that for some items, even though the answer was well known, finding a reference involved citing an obscure manual. For example, one of the references for the 651 exam is *HP POLYCENTER Software Installation Utility Developer's Guide*. Clearly we don't expect everyone to be familiar with developing PCSI kits. So, regarding manuals:

- Don't imagine you can just read all the manuals - you can't!
- If you're already familiar with the manuals, you probably already have the concepts.

- Don't worry if you haven't read them all from cover to cover.
- Reviewing the listed sections is a good idea.

Once you're satisfied that you're familiar with all the concepts listed in the EPG, find the testing center in your area. While taking the exams:

- Read each item carefully.
- Select the best answer(s) presented.
- Be mindful of the total amount of time allowed to complete the exam.
- Rule of thumb is to allow 1 minute per item on a standard test item.
- You can mark an item to review it later.
- Questions will NOT require any complex calculations.
- If in doubt, eliminate wrong answers and guess from the remainder - no penalty for being incorrect.
- "Review marked" and "Review incomplete" at end phase of exam.

What are you waiting for?

With this brief look at the process for creating an exam, I hope I've convinced you that the OpenVMS certification exams are a valid measure of your OpenVMS skills and that certification is a worthwhile goal. Good Luck!

Figure 1 – The goal!

For more information

Refer to the following web site for Information about OpenVMS certification:
http://www.hp.com/certification/

An external course offering intended to prepare for certification can be found at:
http://www.parsec.com/openvms/index.asp?info=openvmstrack1.html

OpenVMS Technical Journal V5

Removing the 32-Bit Limit from System Space (OpenVMS V6.1 -- V7.3-2)

Removing the 32-Bit Limit from System Space (OpenVMS V6.1 -- V7.3-2)

By Dan Buckley, Consultant, HP Services

Overview

With OpenVMS VAX Version 6.0 and earlier systems, 32-bit VAX address space was divided into four 1-GB sections called S0, S1, P0, and P1. S0 space was designated as system space. This 1-GB limit on system space could prevent a system from having a large number of users while also having large working sets, virtual address space, memory disks, nonpaged pool, and LBN caches. If your usage attempted to exceed the amount of system space available, your system would not boot.

As systems grew larger with more CPUs and huge amounts of memory (16 GB is common now, whereas a MicroVAX II was limited to 16 MB), the 32-bit limit prevented systems from effectively using all their memory.

This article tracks the changes made to OpenVMS over the years to expand system space, remove large users of system space, and reduce the size of the remaining users of system space.

Background

With OpenVMS VAX Version 6.0, system space was limited to the 1 GB S0 space, and S1 space was not used by OpenVMS. The largest users of system space were balance slots, nonpaged pool, memory disks, and VIOC cache. Each of these system space consumers is described in the following paragraphs.

Balance Slots

Balance slots use system space to describe all the possible addresses that a process can use. Each balance slot includes space for a working set list of WSMAX entries to describe the maximum physical pages, and page tables of VIRTUALPAGECNT entries to describe the maximum virtual address space. Each balance slot can hold the maximum size process header (PHD). Since large numbers of working set list and virtual page table entries are needed to run a large program in memory without incurring paging overhead, the space required to hold all of these entries can be very large.

On a VAX, the maximum values for the system parameters WSMAX and VIRTUALPAGECNT are 1048576 and 4194304, respectively. With 4 bytes per entry on a VAX, the largest possible balance slot is 4*(1,048,576 + 4,194,304) = 20,971,520 or 20 MB. This means that even in the hypothetical instance where there are no other users of system space, a VAX running Version 6.0 could have only 50 balance slots of the maximum size.

If the number of processes on the system (MAXPROCESSCNT) exceeds the number of processes in memory (BALSETCNT), swapping can occur no matter how much free memory is on the system. OpenVMS VAX Version 6.0 did attempt to reduce this swapping by using virtual balance slots (enabled by the VBSS_ENABLE system parameter), but system slowdowns and hangs were still possible with virtual balance slots enabled if all real balance slots were in use.

Nonpaged Pool

Nonpaged pool is used to hold many OpenVMS and application data structures. These include, but are not limited to, locks, resources, unit control blocks, I/O request packets (IRP), timer queue entries, window control blocks, access control lists, and VAX communication request packets (VCRP). Nonpaged pool is very closely tied to OpenVMS and most likely will never move out of system space.

The system parameter NPAGEDYN specifies the initial allocation of pool on boot and NPAGEVIR specifies the maximum to which nonpaged pool can expand. System space usage for nonpaged pool is one page for each page in NPAGEVIR, so you can gain back some system space on systems where nonpaged pool does not grow by reducing the value of NPAGEVIR to equal the value of NPAGEDYN (or 2*NPAGEDYN). As shown in **Example 1**, you can monitor nonpaged pool usage and expansion by executing the DCL command SHOW MEMORY/POOL/FULL and comparing the current value to the initial size and maximum size. If these values are equal, look at the free space.

```
$ SHOW MEMORY/POOL/FULL
System Memory Resources on 13-AUG-2004 10:15:10.02
Nonpaged Dynamic Memory      (Lists + Variable)
    Current Size (bytes)       2999808    Current Total Size (pages)    5859
    Initial Size (NPAGEDYN)    2999808    Initial Size (pages)          5859
    Maximum Size (NPAGEVIR)   11999744    Maximum Size (pages)         23437
    Free Space (bytes)          553600    Space in Use (bytes)       2446208
    Size of Largest Block        77568    Size of Smallest Block          64
    Number of Free Blocks          426    Free Blocks LEQU 64 Bytes       12
    Free Blocks on Lookasides      331    Lookaside Space (bytes)     128960
```

Example 1 - Using SHOW MEMORY to Monitor Nonpaged Pool Usage

A system's usage of nonpaged pool depends on how the system is set up and how the applications are being run. If you run out of nonpaged pool (expand to NPAGEVIR), the system will crash or hang while trying to allocate nonpaged pool.

Memory Disks

Memory disks (also known as DECram disks) use system memory as a pseudo-disk to greatly increase disk throughputs and decrease file access times. The cost of a memory disk is about 1 page (pagelet on Alpha or I64) of memory per disk block and 1 system page table entry (SPTREQ) for each 16 blocks of disk space under DECram V2.1. Each SPTREQ requires 4 bytes of system space on a VAX, so a large memory disk can consume significant amounts of system space. Putting frequently accessed files onto a memory disk can significantly improve application performance, but large DECram disks can run into system space limits. The use of system space was reduced with DECram V2.2, which uses only 1 SPTREQ for each 16K DECram disk blocks.

Virtual I/O Cache (VIOC)

Virtual I/O cache (VIOC) or logical block number (LBN) cache is a cache of blocks recently read from disk. If another read is made to an LBN that is already in cache, the system does not have to go to disk to get the data, and the associated delay to perform the I/O operation is avoided. The larger the VIOC, the more likely a read hit will occur (some applications do not benefit from read caches), so large caches can produce significant performance improvements. Each VIOC page consumes one page of nonpaged pool and is shown as a VCC packet in the output display from the SDA command SHOW POOL/SUMMARY.

Changes Made to OpenVMS to Address System Space Limitations

OpenVMS Version 6.1 and Later

In OpenVMS VAX Version 6.1 and on all versions of OpenVMS Alpha and OpenVMS I64, the unused S1 address space was combined with S0 space to create a 2-GB system address space called S0/S1 space. VAX machines that do not support extended addressing (XA) remain limited to the 1-GB S0 system space.

OpenVMS Version 7.0

OpenVMS Version 7.0 introduced 64-bit addressing and a new, very large, S2 address space. S2 space defaults to 6 GB, but can be increased by tweaking the SYSGEN parameter S2_SIZE. For more information about the virtual address space layout, refer to Section 1.6 of *OpenVMS Alpha Internals and Data Structures*, Memory Management volume, by Ruth Goldenberg.

Most changes in OpenVMS Alpha (and subsequently in OpenVMS I64) to reduce the use of system space take advantage of these new features by moving data structures from S0/S1 space to S2 space or by moving large users of nonpaged pool to S2 space.

As noted in the following OpenVMS Version 7.0 release note, virtual page tables were removed from balance slots, reducing their size in system space.

Starting with OpenVMS Alpha Version 7.0, VIRTUALPAGECNT ceases to be a tunable parameter on Alpha systems and is no longer used to specify the virtual size of a process. The process page tables have migrated from system space to a dedicated page table space that is guaranteed to be large enough to accommodate the virtual address space provided by the system. This migration has rendered the parameter obsolete, and OpenVMS Alpha ignores its contents entirely.

OpenVMS Version 7.1

Memory resident global sections allow an entire global section to be mapped at boot time, and the pages in the section are not charged against a process's working set. This allows for programs with very large global sections to run on machines with a reduced value for WSMAX, which in turn reduces the amount of system space required for balance slots in system space.

OpenVMS Version 7.2

Lock manager data was moved from nonpaged pool to pool zones in S2 space. Two of the largest users of nonpaged pool prior to Version 7.2 were lock blocks (LKB) and resource blocks (RSB). Moving the data used by the distributed lock manager out of nonpaged pool greatly reduced the amount of NPAGEDYN used on most AlphaServer systems. This reduced pool usage could then be used to lower the value of NPAGEVIR and reduce system space usage.

OpenVMS Version 7.3

Extended File Cache (XFC) in S2 space replaced VIOC in S0/S1 space for LBN cache. Prior to the release of XFC, the largest LBN cache you could make on an AlphaServer was less than 1.5 GB -- no matter how much memory was on the system. With XFC using S2 space, the size of cache is limited only by the amount of memory in the system. LBN caches of 4 GB and larger are common, and these larger LBN caches require no system space.

OpenVMS Version 7.3-1

Preallocated floating-point registers and execution data (FRED) addresses were removed from system space. Prior to this change, either 16 FRED blocks (in Version 7.2) or 256 FRED blocks (in Version 7.2 and later) were reserved in each balance slot at process creation, even if the process did not become multithreaded. Starting with Version 7.3-1, FRED blocks are allocated when a process calls $INIT_MULTITHREADING. Only the primary kernel thread's FRED is in the PHD, thus reducing the size of the balance slots.

OpenVMS Version 7.3-2

Working set list pages were moved to S2 space, reducing the size of each balance slot to 1 page. **Examples 2 and 3** demonstrate the difference in memory usage and the size of balance slots between OpenVMS Version 7.3 and Version 7.3-2. Both examples show output in response to the SDA command CLUE MEMORY/LAYOUT. In Example 3, note the new location of the working set list (WSL), which has been moved outside S0/S1 space, and the reduced size of the balance slots (now just 1 page each).

```
System Virtual Address Space Layout:
------------------------------------
   Item                                Base              End          Length
...
287 Balance Slots, 29 pages each  FFFFFFFF.8366A000  FFFFFFFF.87770000  04106000
```

Example 2 – Memory Usage on OpenVMS Version 7.3 (WSMAX = 196608)

```
System Virtual Address Space Layout:
-----------------------------------
   Item                                 Base              End          Length
System Virtual Base Address       FFFFFEFE.00000000
PFN Database                      FFFFFEFE.00000000  FFFFFEFE.00280000  00280000
188 WSL Slots, 17 pages each      FFFFFEFE.00280000  FFFFFEFE.01B78000  018F8000
Permanent Mapping of System L1PT  FFFFFEFE.01B78000  FFFFFEFE.01B7A000  00002000
...
to the beginning of system space
Execlet Code Region               FFFFFFFF.80000000  FFFFFFFF.80800000  00800000
...
188 Balance Slots, 1 page each    FFFFFFFF.828A0000  FFFFFFFF.82A18000  00178000
```

Example 3 - Memory Usage on OpenVMS Version 7.3-2 (WSMAX = 262144)

Summary

The changes made to OpenVMS Alpha from Version 7.0 through Version 7.3-2 (and included in later versions of OpenVMS Alpha and OpenVMS I64) have made the S0/S1 system space limitation of 2 GB a nonissue. The biggest user of system space is now nonpaged pool, but most of the large users of nonpaged pool have been removed. By setting SYSGEN parameter S2_SIZE to reflect the needs of your system, you are limited only by the amount of physical memory in the system, which can be as high as 512 GB in a 64-processor AlphaServer GS1280. Of course, twenty years from now, folks will shake their heads and mutter "ONLY 512 GB of memory...!" with the same disbelief we have now when contemplating a 16-MB MicroVAX II.

For More Information

OpenVMS Alpha Internals and Data Structures, Memory Management volume, by Ruth Goldenberg (ISBN:1-55558-159-5).

The *Release Notes* and *New Features* manuals for your version of OpenVMS.

About the Author

Dan Buckley joined Digital in 1980 and started working on OpenVMS in 1983 on a VAX 750 running OpenVMS Version 3.2. He has been supporting OpenVMS in the Customer Support Center since 1995.

OpenVMS Technical Journal V5

Taking OpenVMS Security One Step Further

Taking OpenVMS Security One Step Further

Ted Saul, Off-site Software Support Consultant

Michael Grinnell, Off-site Software Support Engineer

Overview

According to Secure Enterprise Magazine, more than two million dollars is lost annually in the United States because of worms, viruses, scripts, and other Internet security problems that disrupt business operations. Larger operations, especially Global 5000 businesses, experience even greater losses. Even though OpenVMS is "virtually unhackable," according to the 2001 DEFcon9 event, lax security methods and irresponsible users can allow OpenVMS system security to break down. It would interesting to determine the amount of money lost in trying to track down these types of security problems that take place on OpenVMS systems.

The four most common security access problems that can be found in OpenVMS environments are as follows:

- User irresponsibility, in which users purposely or accidentally cause noticeable damage.
- User probing, in which users are authorized to access computer resources but use their privileges to exploit insufficiently protected parts of the system.
- User penetration, in which users breach security controls to gain access to a system.
- Social engineering, in which intruders gain access by deceiving legitimate users of the system. Intruders might impersonate users by obtaining access to their unsecured system or terminal or to their passwords, or they might persuade users to perform actions that compromise the security of the system.

OpenVMS provides many internal tools that protect the operating system from harm. However, there are times when it is necessary to take OpenVMS security one step further. These steps can include:

- Monitoring which users are touching particular files or resources.
- Logging what a user is doing with a privilege they have been granted.
- Ensuring that inactive sessions are managed.
- Implementing a stricter set of rules in place for the system.
- Taking and reporting out a snapshot of the current security settings.

When it becomes necessary to take these and other severe security precautions, a product from one of HP's partners can help. PointSecure, Inc., provides OpenVMS IT managers with three products that can help with the management of these and many more tasks.

PointAudit

What is PointAudit?

The PointAudit product is a vulnerability assessment tool designed to check certain profile, privilege, file, and system parameter settings, and to alert you to exposures that might exist on your OpenVMS system. Currently, there is no way on an OpenVMS system to get an "at-a-glance" check of security settings other than by looking up each setting individually. PointAudit provides this ability and provides a series of customizable reports that allow you to see how secure your system is.

How does PointAudit work?

PointAudit resides on a Windows® based server and pulls its data from the OpenVMS system. PointAudit stores this information in a database on the Windows server for reporting and for comparing against subsequent data captures. Before the first data capture can take place, a few required steps must first be performed. First, your company's security policy information is translated into settings that PointAudit can use for comparison of your OpenVMS systems. For example, required password lengths for system and nonsystem accounts are set, as are their respective expiration ages. Second, default values for SYSGEN parameters, UAF flags, privileges, and granted quotas are also recorded. Once these values are in place, the desired data and output formats can be generated. Finally, all the settings are saved to the PointAudit database on the Window server and for use in future analysis.

PointAudit performs a variety of tests, which are divided into four categories:

- User profile tests
- Privilege tests
- File checkup tests
- Parameter tests

The **User profile test** and the **Privilege test** analyze the SYSUAF.DAT file. These tests check for the following:

- Users within the DEVOUR privilege group
- Users within the SYSTEM privilege group
- Users within the OBJECT privilege group
- Users within the ALL privilege group
- Users whose password length is too short
- Users with passwords that never expire
- Users who haven't changed their password within a specified time
- Users with failed login attempts
- Users whose UIC is not found in the rights database

This data can then be compared against the company security policy that was replicated during setup of PointAudit. For example, if a 15-character password is required by the company, violations of this rule will show up on the report. (A later section of this article discusses the System Detective application, which can identify users who are using their privileges to set up accounts that bypass security policy.)

The **File checkup test** analyzes stored files and reports potential security flaws. This information includes:

- Files containing an invalid general identifier
- Files containing an invalid UIC identifier
- Files containing a wildcard identifier
- Files containing an invalid owner
- Files with world read and world delete access

OpenVMS does allow each of these settings; however, they might be considered a security breach in some environments. For example, files that do not have a valid owner might indicate that a user was deleted from the system without proper cleanup of their files. If a new user is then added to the system and the old UIC reused, the new user will immediately have access to the previous owner's

files. Depending on what is recorded in these files, this situation could represent a serious security breach.

Parameter tests look at login, system, and network SYSGEN parameters to determine whether the environment is as secure as possible. This test looks at specific parameters related to login security, including LGI parameters, and compares them against the recommended values list. Those that do not match are flagged and reported as potential security breaches. Parameters that affect system and network security are also checked and compared. (The comparison values are set during the initial configuration.) The test also reports information about installed licenses and products as well as installed and removed patches. This information is useful in determining a patch level for a particular server and whether any attempt has been made to change the performance level of the machine.

Most of the PointAudit tests can be run either as a group or individually. A GUI interface on the Windows server indicates when each test has run and whether or not it was successful. The number of users and files that have met the test criteria is also reported, as is an immediate indication of security risks.

How does PointAudit enhance OpenVMS security?

The PointAudit product enhances OpenVMS security by allowing for the quick viewing of critical security settings. This "at-a-glance" view eliminates the need for issuing commands like DIR/SEC throughout the system and entering multiple SYSUAF and SYSGEN commands, thereby quickly uncovering violations of the security policy. By running tests and saving information over a period of time, changes that might compromise the system are discovered much more quickly, thereby allowing for a faster reaction time. A baseline for almost every aspect of the system's security can also be set for use in this quick comparison. PointAudit can perform this comprehensive security check in a short amount of time and can give an immediate indication of whether unauthorized personnel have been granted potentially damaging privileges.

PointSecure System Detective (Automatic Operations)

What is System Detective?

The PointSecure System Detective application runs on an OpenVMS server in the background and performs three primary functions:

- Monitors user activity.
- Manages inactive sessions.
- Provides access management and protect against intrusion.

With the use of screen messages, log files that contain details of user's session keystrokes, and database searches, the OpenVMS system administrator can see what is currently taking place on the system. They can also look back to view what has taken place previously. System Detective automatically takes action when certain events take place on the system.

How does System Detective work?

To monitor user sessions, rules are set up to report activity. These rules can be created to watch for certain behaviors, including the following:

- When a user logs on and takes certain actions
- What a user is doing and has done on the system
- Who a user is and to what security group they may belong
- Why a user may be carrying out a particular function such as enabling a privilege

System Detective can be set to turn on monitoring when a particular action takes place in order to watch a user or even to disable that user. Conversely, monitoring can be turned off when another event takes place. For example, a system administrator might want to turn on monitoring when a user invokes the Authorize utility (AUTHORIZE). All keystrokes can be recorded to a file, thereby enabling an audit trail if suspicious behavior occurs. When the user exits AUTHORIZE, monitoring can be turned off and all recording stopped.

When monitoring turns on, two actions take place:

- Every keystroke is recorded to a log file. This log file can be reviewed during the session or at a later time.
- The System Detective database records that a monitor event has taken place so that the system administrator can be notified.

Setup of System Detective begins with the editing of a configuration text file. Three types of commands are available to define the rules by which the system policy is defined:

- **Action commands** tell System Detective what to do when a particular event occurs on the system. Some of these commands are:
    - Monitor_User – Tells the automatic operations when, what, who, and why a user should be monitored. When this action takes place, keystroke collection and event monitoring to the database can begin.
    - Disable_User – Manages inactive sessions that have been detected on the system. Actions that can be defined include issuing warnings, and locking, suspending, or even terminating sessions.
    - Lockout_User – Sets the rules for locking an active user out of the system. The system administrator has the choice of simply suspending the user, of warning an offending user, or of terminating them altogether. Details of the action taken are then recorded in the Automated Operations database.

- **Global commands** direct System Detective and allow the system administrator to define commands to be used either throughout the entire configuration file or only in certain sections of the file. Global commands can also define values to be used by other commands. Commands available include:
    - Exclude_User – Exempts a particular user from actions take by other commands. For instance, you might want to exempt the SYSTEM account from being locked out for any reason.
    - No_Suspend – Prevents System Detective from being locked out for any reason.
    - Set_User_Class – Defines a group or class of users. For instance, you might want to group certain users together for monitoring purposes and to eliminate the need for listing separate commands in the configuration file.
- **Supplementary commands** include the following:
    - Report_User_Event – Records all user logins and batch job creations.

- Session_Lock – Manages the session lock utility, which provides users with a method of locking their terminal.
- Disable_DEC_Windows – Enables inactivity management for DECwindows workstations.

To enhance each of these commands, a series of objects, object types, options, and parameters are available. The use of these modifiers allows OpenVMS system administrators to integrate their system security policy into System Detective for automatic monitoring of systems. Such modifiers provide the in-depth granularity that allows monitoring of every aspect of the system.

Finally, the SYSDET command is also available to system administrators. Used from DCL level, SYSDET has the ability to start and stop System Detective as well as to manage licensing and reporting. Batch processing can also be used to take action when a certain event takes place on the system. This functionality is useful for sending email or for setting off a pager to notify the system administrator of security breaches as soon as they take place.

How does System Detective enhance OpenVMS security?

OpenVMS has the ability to put the first level of defense to protect its operating environment. By itself, OpenVMS has proven to be quite effective. System Detective, however, gives the system administrator the ability to guard against specific attacks and to take appropriate action against specific threats. Because most OpenVMS attacks typically come from within the system, automated operations can aggressively protect against and alert when intrusions or privilege abuses take place. OpenVMS does take action against security threats such as suspected break-ins and login failures. The addition of System Detective allows the careful monitoring of all users who are logged in to the system as well as the logging of precise actions that are taking place. In this way, the four vulnerable areas of an OpenVMS system can be closely monitored and managed in an unattended environment.

Using System Detective also provides the system administrator with a better understanding of how to use OpenVMS security internal features. Because of the depth of monitoring provided by the product, the system manager must be extremely familiar with the security settings of their system. Functional areas of the system that are not normally monitored might be exposed and would be served well by the use of System Detective. For example, the product can reduce the chance that privileges granted to a user is accidentally forgotten about or that an improper file protection is left in place on a critical image. System Detective helps to organize all security-related changes, thus allowing even the most inexperienced system administrator a good understanding of their system.

HP and PointSecure Services

Two key security concerns face OpenVMS system administrators, security administrators, and IT directors:

- Managing a system that requires elevated privileges for programmers, operators, third-party consultants, and vendors. The more privileges that are granted, the greater the potential security risks.
- An increased demand from internal and external audit requirements, such as HIPAA, Sarbanes-Oxley, and Graham-Leech-Bliley. Companies are being forced to build strong security polices in order to comply with new laws, internal pressures, and more stringent internal auditing demands. It is vital that customers can prove they are keeping their clients' data private.

HP has joined forces with PointSecure, Inc., to offer the OpenVMS System Security Audit Service. The goal of this service is to provide customers with:

- A comprehensive understanding of their OpenVMS security settings
- A solution for continued security management and system auditing

The OpenVMS System Security Audit Service is sold on a per-node basis and is based on tier (Workgroup, Departmental, or Enterprise). A single component or a combination of components can be chosen:

Option 1: Active Audit provides a snapshot security audit and installation of the PointAudit software.

Option 2: Proactive Audit provides a snapshot security audit and the installation and configuration of the System Detective monitoring environment.

Option 3: Ultimate Audit provides a snapshot security audit, installation of the PointAudit software, and installation and configuration of the System Detective monitoring environment.

Key Benefits of the Service

Customers can benefit from the OpenVMS System Security Audit Service in several ways:

- The security audit provides an HP-certified report of the OpenVMS system that compares current security level against best industry security practices.
- The installation and configuration of PointAudit includes a license and maintenance contract for one year of telephone support.
- System Detective provides the ability to proactively enforce security policies, along with the license and maintenance telephone support contract.

Summary

With the increased concern about security, it is imperative that system administrators take the most proactive steps possible to secure their systems. With the help of PointSecure and HP, even OpenVMS -- the most secure operating system in the world -- can have a more secure environment and can also be managed easily. If an audit from the PointAudit software doesn't report the same level of security you thought you had, then it might be time to consider deploying these products on a full-time basis.

For more information

For more information about PointSecure products, visit http://www.pointsecure.com. For more information about the HP OpenVMS System Security Audit Service, visit http://www.hp.com/hps,

OpenVMS Technical Journal V5

DECnet-Plus Technical Overview

DECnet-Plus Technical Overview

Author: Colin Butcher, Technical Director, XDelta Limited

Overview

This article discusses DECnet-Plus (formerly known as DECnet/OSI), the current implementation of DIGITAL Network Architecture (DNA) Phase V for OpenVMS.

Today's DECnet networks support remote system communication, resource sharing, and distributed processing. Network users can access resources on any system in the network as well as the resources of other vendors' systems on multivendor networks.

All systems connected to a DECnet network are peers or equals. Systems can communicate with each other without having to traverse a central or master system. Any system can communicate with any other system in the network via specialized devices such as routers, not just to those systems to which it is directly connected.

This article includes:

- A historical perspective of the development of DECnet, from the initial rudimentary networking protocol for small numbers of similar computers to the current DECnet-Plus that embodies the Open Systems Interconnection (OSI) standards and protocols, enabling support for an unlimited number of heterogeneous computers in a multivendor, multiprotocol network
- Descriptions of the main concepts and components of DECnet-Plus, including the OSI layered architecture, Phase V command line interface, node name and address resolution, node addresses and address towers, routing, time synchronization, complementary protocols (such as MOP), and DECnet over TCP/IP
- Brief guidelines and recommendations for choosing implementation and installation options, including advantages and disadvantages where applicable

The main purpose of this article is to provide users and system managers a greater understanding and appreciation of the behavior and capabilities of DECnet-Plus. The author has extensive consulting experience working with DECnet on OpenVMS systems and has written this article in response to questions that arose from users, programmers, and system administrators.

What Is DECnet?

DECnet is the underlying set of rules and software components that enable a wide variety of computer systems to exchange data safely and reliably. The rules describing and enforcing the behavior of the DECnet protocol are carefully constructed and documented in the DIGITAL Network Architecture (DNA) specifications. Each system (or **node**) participating in the DECnet network must rigorously adhere to the rules to ensure consistent and reliable communication with other nodes in the network.

DECnet-Plus is the most recent implementation of DNA Phase V. DNA Phase V incorporates the Open Systems Interconnection (OSI) communications specifications as defined by the International Organization for Standardization (ISO) and maps onto the OSI seven layer reference model.

DNA Phase V also specifies the mechanisms by which a DECnet-Plus (or earlier DECnet/OSI) implementation can use the TCP/IP protocol stacks as a carrier (implemented as HP TCP/IP Services for OpenVMS). This allows existing DECnet applications to operate unchanged in an IP only infrastructure by preserving the end-to-end application programming interfaces (APIs). This functionality is commonly referred to as **DECnet over IP**.

In any network, the protocols for transferring data between systems need to control several major levels:

- The physical level, such as hardware interfaces and cabling
- The interchange level, such as data flow control, integrity checking, and retransmission
- The routing level, such as node addressing and optimal path determination

- The user level, such as the command line interface and application programming interface

Basic data exchange mechanisms such as the Kermit file transfer program implement a simple point-to-point connection. By contrast, a complex heterogeneous data network protocol such as DNA Phase V (where many computers can simultaneously exchange data with many others) requires a far more rigorous approach to design and implementation.

The rules enforced by the Application Programming Interfaces (APIs) in a DECnet network serve to isolate the user of a service from the lower-level details of the network (the physical and interchange levels). If the rules that govern the external (outside the system) application-level programming interface stay consistent, and if all systems run compatible versions of the DECnet protocol, then these systems can exchange data while running any version of the operating system on any hardware platform.

This independence of layers facilitates modifying the network — for example, replacing an OpenVMS VAX V5.5-2 system running DECnet Phase IV with an OpenVMS Alpha V7.3-1 system running DECnet-Plus (in Phase IV compatible addressing mode, which is discussed later in this article). No changes to the application code are needed. The other nodes in the network do not know and do not need to know that the replaced node is now a physically different system. The application software only "sees" the corresponding (unchanged) application on each node through the DECnet end-to-end connection.

A consistent set of APIs allows the design and implementation of "network aware" applications that can be distributed over as many systems as necessary to provide the necessary scalability. One essential concept is that of a network connection to the same node as the originator. This allows network aware applications to be built and tested on physically small networks, then delivered onto larger distributed networks. Realistic testing is essential to ensure that problems of scale and performance do not arise in production use.

What is **DECnet Phase IV**? As described in more detail in the next section, DECnet Phase IV (also known as DNA Phase IV) is one of five major phases in the development of the DECnet protocol. DECnet Phase IV is the DECnet that most people are familiar with. It introduced support for a large number of nodes (up to 64,449 compared to Phase III's maximum of 255) and added support for area routing. DNA Phase IV was first implemented as DECnet Phase IV, then became DECnet Phase IV-Plus when two significant features were introduced: end node failover and out-of-order packet caching.

> **End node failover** allows a Phase IV End Node to be connected to two entirely separate circuits, but using only one circuit at any one time with automatic and transparent failover from one circuit to the other. The "standby" circuit is entirely operational, actively listening to routing updates and maintaining the node reachability data. However, it is not used for transmitting application data until the "primary" circuit has failed.
>
> **Out-of-order packet caching** allows a Phase IV Routing Node to load balance over multiple equal-cost circuits. Prior to Phase IV-Plus, load balancing over multiple available paths to a destination node was not possible. The out-of-order packet cache feature solved the underlying issue: with multiple available paths between nodes, packet arrival at the destination could not be guaranteed in the same order as the packets were originally transmitted. In contrast, a single path between nodes implicitly guarantees that packets arrive in the same order as they were transmitted.

Host-based routing was part of Phase IV but was not implemented in the release of DECnet/OSI, which required the use of dedicated external routers (such as DECnis and RouteAbout). Host-based routing was re-introduced into Phase V with DECnet-Plus and OpenVMS V7.1. **Host-based routing** allows an OpenVMS system to route data from a local area network (LAN) through a wide area network (WAN) that needs a separate dedicated router. In Phase V terminology, a node running host-based routing is an **OSI Intermediate System (IS)**. Note that it is entirely valid to have a routing node with a single path to provide routing updates about node reachability to other nodes on a LAN.

The Phase V equivalent of a Phase IV End Node is an **OSI End System (ES)**. Phase V nodes with multiple paths are referred to as **Multi-Homed systems**. They each include an out-of-order packet cache. Multi-Homed End Systems can thus load balance over multiple available paths.

The DIGITAL Network Architecture Phase V specification is published and can be purchased if required. Other protocols in the lower layers of the DECnet networking hierarchy also conform to their specific architectural specifications. For example, Ethernet (in all its variants) and the physical cabling such as Category 5 Twisted Pair conform to their own architectural specifications.

The DNA Phase IV specifications are available on the Internet.

Some of these specifications are made available under license, such as LAT (Local Area Transport), which is typically used by Terminal Servers (DECservers) to exchange serial terminal traffic with a host computer, typically over Ethernet.

The RFCs for TCP/IP constitute a set of specifications for individual components of the TCP/IP protocol suite.

A Brief History of DECnet

DECnet was originally designed and developed by Digital Equipment Corporation (DEC), whose outstanding architectural approach and engineering excellence laid the foundation for many modern software and hardware developments. DECnet has evolved over the years from a simple protocol (DNA Phase I) connecting two computers over an RS232 serial line to a sophisticated and complex protocol (DNA Phase V) capable of interconnecting a virtually unlimited number of systems, including those from other manufacturers. The DEC internal network (EasyNet) was probably the largest DECnet network to exist in terms of the number of connected nodes.

Over the past few years, DECnet has been overtaken by TCP/IP as the preferred means of interconnecting systems. To some extent, this has been driven by a simplistic approach to the provision of backbone networks by "managed service" suppliers whose networks are built on equipment from the major network hardware vendors. The majority of those vendors only provide TCP/IP routing, with some providing DECnet routing functionality at additional cost.

A number of vendors provide high bandwidth, low latency "layer 2 services" that give customers a greater choice of protocols to use. These are ideal for applications such as split-site clustering, in which OpenVMS uses the SCS cluster protocol and fibre-channel based storage subsystem interconnects.

DECnet Phase V implements the OSI 7 layer reference model for vendor-independent computer inter-networking. All the Phase V network management entities map clearly onto the OSI 7 layer model (this model is discussed in more detail later). One of the main design issues behind the introduction of Phase V was the need to increase the available address space to accommodate more nodes in the entire network, as well as to manage the routing updates for all the nodes. This need was elicited initially by the growth of the EasyNet DECnet network.

Each new version of the DECnet architectural specification retained backward compatibility with the previous version, thus facilitating migration from one version to another. Initially, DECnet was associated with the RSX and VAX VMS operating systems as the primary network mechanism for interconnecting such systems. (RSX is a PDP-11 operating system and the cultural predecessor to VMS; VMS is the original name of the OpenVMS operating system.) Each version of the network layer specification was linked with a particular version of the OpenVMS operating system; however, recent versions of the DECnet architecture have become independent of the OpenVMS operating system version, to the point where DECnet is no longer a prerequisite for installing and configuring OpenVMS (V7.1 and later).

Incidentally, this separation of DECnet from OpenVMS led to the need for other mechanisms for loading network-booted devices such as cluster satellites and terminal servers. This inspired the introduction of the LANACP and LANCP components, which provide a DECnet-independent method of servicing MOP load requests.

Figure 1 shows the relationships between the versions of OpenVMS and DECnet, from VAX VMS V4.0 with DECnet Phase IV to OpenVMS Alpha and I64 V8.2 with DECnet Phase V:

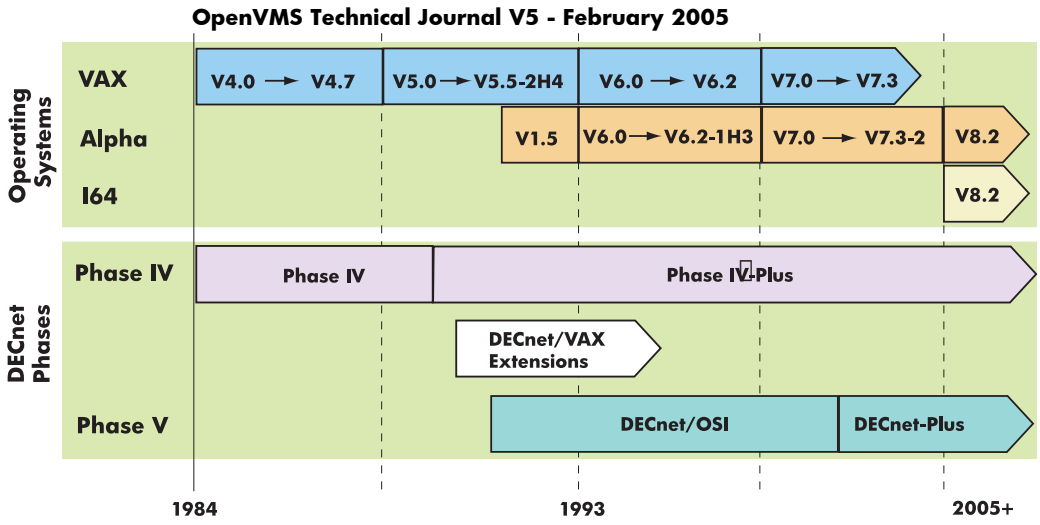

Figure 1 OpenVMS and DECnet Versions

During the evolution of the DECnet protocol, the inter-computer links themselves evolved rapidly in response to the changes undergone by the hardware technologies. One very important aspect of a layered network architecture design is that the end-to-end protocol for the exchange of data need not be changed when the lower layers change (such as the physical medium layer). For example, such a design approach allows you to replace a serial line with Ethernet and to subsequently evolve to gigabit Ethernet – all this without requiring changes to the upper protocol layers.

Another example demonstrating the importance of a sound architectural specification is the ability of the lower layers to operate today at many times the speed of the originally designed lower layers, and yet the old and new can co-exist without creating significant timing issues. Both Phase IV and Phase V follow the principles of a layered design with tightly defined and enforced interfaces between layers.

Table 1 lists some of the major changes and features introduced with each phase of DECnet:

Phase I	Device-specific point-to-point link between computers using low-speed wide area network (WAN) style serial links; limited to two computers.
Phase II	File transfer, remote file access, task-to-task programming interfaces, network management for point-to-point links; up to 32 computers.
Phase III	Adaptive routing (indirect connections between computers), downline loading, record level file access; up to 255 computers.
Phase IV	Ethernet local area network technologies, area routing, host services; up to 64,449 computers.
Phase V	OSI protocol support, transparent transport level links to TCP/IP, multivendor networking, local or distributed name service, distributed network management; virtually unlimited number of computers.

Table 1 The DECnet DNA Phases

Note that Phases I to III predate LAN technologies such as Ethernet.

Phase I was very basic and provided a device specific point to point link between computers using low speed WAN-style links.

Phase II increased the scale of the network, but still required point to point links between the computer systems.

Phase III increased the scale yet again, introducing the concept of routing (indirect connections between systems).

Phase IV increased the scale of the network significantly from the earlier phases, introducing a new address space to enable support of over 64,000 nodes and area routers to reduce the scope of the

routing update problem incurred in earlier networks as they grew in size. Phase IV provides areas in the range 1 to 63 and addresses within each area in the range 1 to 1023, thus giving a total of 63x1023 possible node addresses on a single interconnected network.

Phase IV provides both End Node and Routing Node (layer 1 "within area" or layer 2 "between areas") functionality. Originally, DECnet Phase IV was embedded in the OpenVMS system (Version 4.*x*, 5.*x*, 6.*x*, and 7.0) as the default network mechanism. For many years it formed the backbone of the Digital internal network known as EasyNet. Beginning with OpenVMS V7.1, DECnet Phase IV was no longer embedded as the default network protocol and became available instead as a separate layered product in the OpenVMS distribution kit.

Phase V was introduced primarily to solve addressing scheme limitations brought to light with the rapidly growing internet and also to provide full OSI networking for interoperability with other vendors systems. Phase V also provides full interoperability with Phase IV which is essential for implementing a phased transition from Phase IV to Phase V throughout an organization. Phase V was initially released in a number of stages such as Wave 1 and Wave 2; for example DECnet/VAX extensions for DECnet Phase IV.

Phase V came of age around the time of OpenVMS V6 with the release of DECnet/OSI V6. At this point, DECnet became a much better integrated and packaged product, particularly with the introduction of a local DNS-based local node database.

Phase V was initially implemented with OpenVMS systems acting only as OSI End Systems (End Node equivalent) and with dedicated hardware routers acting as OSI Intermediate Systems (Routing Node equivalent). This was due to both a purist approach to the overall design and to the performance restraints executing the Phase V routing code. With the introduction of a new, unfamiliar (but powerful) network management interface and the hardware expenditure and cabling disruption required for an upgrade from Phase IV to Phase V, most customers stayed with the simpler and more easily understood Phase IV version of DECnet. Phase IV met most of these customers' requirements.

However, with the advent of the very powerful OpenVMS Alpha systems, host-based routing with OSI Intermediate Systems became feasible and was eventually re-introduced into OpenVMS to provide OSI IS-IS functionality. This eliminated the requirement to have dedicated hardware router devices, although from a network design viewpoint they may be preferable. The re-introduction of host-based routing allows the majority of DECnet users to migrate to DECnet-Plus. It also allows the use of small, inexpensive systems (such as DS10 and RX2600) as dedicated routers for both DECnet and TCP/IP protocols. These systems can also provide other network services, such as MOP downline loading and time servers.

The Layered Approach

The OSI Reference Model, otherwise known as the OSI Seven Layer Model, describes the various layers involved in networks such as DECnet-Plus. The purpose of the OSI Seven Layer Model is to enable dissimilar systems to communicate with each other, the objective being an open architectural specification capable of implementation by many different vendors, thus enabling a high degree of interoperability. Table 2 describes the seven layers of the OSI Reference Model:

Layer	Name	Purpose
Upper Layers		
7	Application	Supports application-specific and end-user processes. Provides for distributed processing and access, contains application programs and supporting protocols (such as FTAM).
6	Presentation	Maintains independence from differences in data representation by translating from application to network format and vice versa. Coordinates conversion of data and data formats to meet the needs of the individual applications.
5	Session	Organizes and structures the interactions between pairs of communicating applications. Establishes, manages, and terminates communication sessions between the applications.

Lower Layers		
4	Transport	Provides reliable transparent transfer of data between end systems with error recovery and flow control.
3	Network	Enables communication between network entities to provide switching, routing, forwarding, congestion control, error handling, packet sequencing, and so forth.
2	Data Link	Specifies the technique for moving data along network links between defined points on the network and how to detect and correct errors in the Physical layer (layer 1).
1	Physical	Connects systems to the physical communications media and transfers the data (bit stream) at the electrical and mechanical level.

Table 2 The OSI Reference Model Layers

The upper layers (layers five to seven) are often referred to as the application layers. The lower layers (transport layers) are typically implemented by the network infrastructure components (repeaters, bridges, switches, routers, and so forth). DIGITAL Network Architecture Phase V is based on this layered structure.

The layered model splits the elements of a network into interdependent layers, where each layer exchanges information with its corresponding layer on another system by means of using the underlying layers. Each layer provides a service to the layer immediately above it, and each layer is supported by all the layers beneath it.

As shown in Figure 2, the OSI layered model data is passed down the layers on one system from the application layer, across the physical connection at the bottom layer (layer one, the physical layer) and then back up the layers on the other system to its corresponding application layer. The vertical arrows show the actual flow of data between the layers of a node (inter-layer data exchange). The horizontal arrows reflect the resulting communication or understanding between corresponding layers (peer level data exchange) of the two nodes.

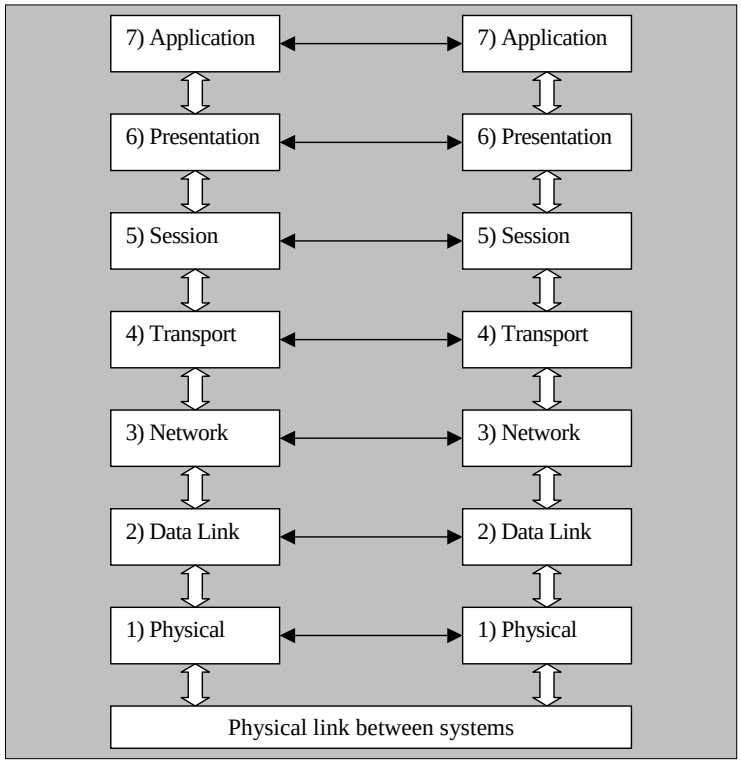

Figure 2 Data Flow in the OSI Model

DIGITAL Network Architecture Phase V

The remaining sections of this article focus on some of the major features and facilities provided by DECnet-Plus.

The Phase V Command Line Interface (NCL) and How It Differs from the Phase IV Interface (NCP)

One of the most important and noticeable differences between Phase IV and Phase V is in the command-line interfaces. DECnet Phase V introduced the Network Control Language (NCL) command interface, which replaces the DECnet Phase IV Network Control Program (NCP) command interface. The different layers in DECnet-Plus are much better structured for management purposes. The NCL entity hierarchy and syntax reflects the DECnet-Plus internal structure of the network management components, neatly mapped onto the OSI seven layer reference model.

Entities are the manageable components that make up the network; they relate to other entities on the same system. For example, the topmost entity in the management hierarchy is the node, which is identified by a globally unique node name. Next below that are various local entities (or child entities also referred to as module entities) such as Modem Connect Line, OSI Transport, DIGITAL Data Communications Message Protocol (DDCMP), and High-Level Data Link Control (HDLC).

There is only one instance of each of these module entities, so each can be identified uniquely. Each module entity has subentities below it. The HDLC module, for example, maintains HDLC LINK entities for each communications link over which the protocol operates; *hdlc link* is the full class name of the communication link entity. Each instance of the HDLC LINK entity requires further identification to allow it to be distinguished from the others. For example, in HDLC LINK HDLC-1, HDLC-1 is the instance name that further identifies the HDLC LINK entity.

NCL directives, or commands, let you manage DECnet-Plus entities by means of their unique network entity names. Unfortunately, users familiar with NCP might find the NCL interface verbose and difficult to grasp quickly; however, with a little patience and some of the helpful migration tools available, users will find that NCL is far more useful and powerful. Two tools that help users learn to use NCL include DECNET_MIGRATE in SYS$UPDATE, which provides approximate NCP to NCL equivalents, and NET$MGMT in SYS$SYSTEM, which provides a DECwindows interface and has an option to show the generated NCL commands.

Another major network management difference introduced with Phase V in DECnet-Plus is the management of network configuration databases. Phase IV network management involves a volatile and permanent configuration database. The volatile database stores active configuration values reflecting current conditions. They are in effect only while the network is running; they are lost when the network is shut down. The permanent database stores the initial values for configuration parameters, and these values are used when the network is started up. (Default values are embedded into the DECnet software and are overridden by changes to the permanent and volatile databases.) Changes made to the permanent configuration database remain after the network is shut down but do not affect the currently running network. With Phase IV management, volatile configuration values are modified with the NCP SET command, while the permanent configuration database values are set with the NCP DEFINE command.

Phase V performs all configuration actions at network startup by using NCL scripts to configure management entities as the network images are loaded and configured. It is helpful to regard these NCL scripts as constituting the permanent configuration database. The NCL scripts control all network entities except for node naming and addressing. You can edit these scripts with a text editor. Be careful not to make inadvertent changes. It is recommended to set the file version number to ;32767 to prevent inadvertent creation of new versions of NCL script files.

The NCL scripts do not set every single value for every single parameter associated with every single entity. The network software as shipped contains default values for many of the parameters. The actual values can be seen with the NCL SHOW *entity-name* ALL command.

Beware that these defaults may change from version to version of the released software (as indeed happens with OpenVMS itself, such as the recent changes to some parameter defaults in OpenVMS V8.2). Read the manuals and release notes carefully and try not to specify everything in the NCL scripts. As a general rule, it is best to use the default parameter values. Change them only if necessary, and only make the minimum changes necessary once you understand their effects and implications.

Each entity has several attribute groups: characteristics, counters, identifiers, and status attributes. Characteristics are the attributes you can modify, although not all of them are modifiable. Identifiers are set when the entity is created, and can be modified only by deleting the entity and recreating it with a new identifier. Note that changes to certain entity characteristics (static versus dynamic parameters – as with DECnet Phase IV or OpenVMS itself) do not take effect until the entities are restarted or if the system is rebooted.

You can also modify entities "on the fly" using NCL commands interactively (as done with NCP commands). This is useful for temporary changes to the running node, where you modify the in-memory active network software entities (as in the Phase IV volatile database). The changes become effective immediately but last only until the system is rebooted. For example, you might want to monitor a set of counters for a particular entity or you might want to temporarily disable a link.

NCL permits constructs such as WITH, which makes commands much more flexible, powerful, and useful. For example, in the following command the use of the WITH construct allows you to limit the display of session control port parameters to those with a specific value for the process identifier:

SHOW SESSION CONTROL PORT * WITH PROCESS IDENTIFIER = "PID_VALUE"

You can use this type of construct with almost all NCL commands. You can experiment to find commands that are most useful for your own circumstances. For example, try to find all currently available nodes on the LAN by asking the nearest router what active Phase IV-compatible addressing end systems it can see (hint – look for adjacencies on the routing circuits).

For more information on DECnet-Plus network management, refer to the *DECnet-Plus for OpenVMS Network Management* manual. For more information on NCL, refer to the *DECnet-Plus Network Control Language Reference* manual. NCL online help (using the NCL HELP command) is also extremely useful, especially with DECnet-Plus V7.3 and later.

The following are commonly useful entities and other manageable objects in the NCL command hierarchy:

- **Implementation** – The read-only Phase V implementation name and version as embedded in the software.
- **Node** – The local node addressing data. One node entity exists for the node module, and it crowns the hierarchy represented in the entity model described by the DNA specification. All other entities are subordinant to the node entity.
- **Session control application** – A network object (to use Phase IV terminology) that manages the session layer, negotiating and establishing connections.
- **Session control port** – The actual software link to the given application. The session control port stores session control information about the transport connection. One of the values is the corresponding **transport port**, which will be either an NSP port or an OSI transport port (see **NSP port / OSI transport port** below). Another value is the OpenVMS **process identifier** of the active process. This enables you to track the amount of network traffic each process is passing to each node.
- **Routing** – Manages the routing layer. Routes messages in the network and manages the message packet flow.
- **Routing circuit** – A data link or path to another node, available to the routing layer over a CSMA/CD station or DDCMP/HDLC link logical station.
- **NSP** – The Phase IV Network Services Protocol transport layer, it implements one of the protocols in the DNA transport layer.
- **OSI transport** – The OSI transport layer, which implements the OSI Connection-Oriented Transport Protocol specification (International Standard ISO 8073) on DECnet-Plus for OpenVMS.

- **OSI transport template** – The template available for use by the OSI transport layer; it supplies default values for certain parameters that influence the operation of a port on a transport connection. The two main templates are RFC1006 and RFC1006Plus (RFC1859). They implement the interface between the OSI transport layer and the TCP/IP ports 102 and 399 using the PATHWORKS Internet Protocol (PWIP) driver. This enables TCP/IP to be used as part of the transport layer for both OSI and DECnet communications between nodes.
- **NSP port / OSI transport port** – The actual software link over the given (NSP or OSI) transport. This is most useful for detecting what traffic (if any) is passing over a specific software connection. One of the values is the corresponding **session control port** (see above).
- **DDCMP / HDLC link** – A link using a port for the given (DDCMP or HDLC) protocol. DDCMP is the Digital Data Communications Message Protocol used over an asynchronous serial line (not supported by DECnet Phase V on OpenVMS VAX). HDLC is the High-level Data Link Control protocol, generally used over a synchronous serial line or frame relay.
- **DDCMP / HDLC link *xxx* logical station** – A specific connection over a DDCMP or HDLC link. It is useful for multi-drop links where a point-to-point link is a special case of a multi-drop link (for example, RS449).
- **Modem connect line** – The physical connection of a port used for DDCMP, HDLC, or a similar protocol. It is not applicable to CSMA/CD LANs.
- **CSMA-CD station** – A LAN adapter. CSMA/CD is the Collision Sense, Multiple Access, Collision Detect LAN protocol mechanism that includes LAN protocols such as Ethernet.
- **DTSS** – The DTSS server and clerk for synchronizing and managing system clocks in the network.
- **MOP** – The MOP (Maintenance Operations Protocol) database for downloading boot images over LANs.

Node Name and Address Resolution

DECnet Phase V implementations support several name services for storing and mapping node names and addressing information: the Local namespace, DECdns (DIGITAL Distributed Naming Service), and DNS/BIND used for DECnet over IP (also referred to as DOMAIN by DECnet-Plus software). Using NET$CONFIGURE, you can configure the naming lookup function to use any or all of the available name services in any specific order (you must select at least one). When you use more than one name service, the search list defines the order in which the existing services are to be accessed.

DECnet node names and address towers are maintained in DECdns and the Local namespace. These are managed using the DECNET_REGISTER utility.

TCP/IP host name and address information is maintained in DNS/BIND or in the local HOSTS database. These are managed using TCP/IP utilities.

The Local namespace (or Local Naming Option) is the most simple DECnet node naming mechanism. The Local namespace is similar to the permanent node database (NETNODE_REMOTE.DAT) used on DECnet Phase IV systems. All values are stored in a single data file on the local node and accessed similarly to how DECdns accesses names. The Local namespace can support up to 100,000 nodes. It works well for relatively small and static networks (10s or 100s of nodes in a well defined network that is not changing or expanding rapidly).

All node names in the Local namespace are prefixed with LOCAL:. to indicate the use of local naming. One main advantage of using the Local namespace is that it provides fast name-to-address lookups (it does not have to interrogate the nearest available name server as do other name services). In addition, no extra software is required for using this name service. The main disadvantage is you have to administer name-to-address mapping information separately on each node, and you must keep all nodes concurrently updated with the local naming database. This is not a significant problem for most networks, especially for stable networks in which the node population rarely changes.

DECdns provides a network-wide distributed database of node names for node name-to-address translation. DECdns is implemented as a global service accessible by any client node on the network. It ensures consistent network-wide name and address information. DECdns requires at least one and

preferably a minimum of two nodes configured as DECdns servers. DECdns behaves in a similar manner to local naming except that node population changes can be made centrally at the DECdns servers, which will in turn automatically propagate the changes to all nodes in the network. Note that the use of DECdns can impose additional connection time when first establishing a network link. This is because establishing the connection requires a DECdns lookup if the name resolution data is not cached locally or the data has not been updated and propagated (thus invalidating the local cache).

You should establish at least one DECdns server per LAN in a large WAN interconnected network. This minimizes DECdns lookup times by other nodes on the LANs. One potential problem with DECdns is that having a single master read/write server and several read-only copies of that server can lead to vulnerabilities due to the single point of failure. If the single master read/write server fails, then updating DECdns node-address information might be temporarily impossible — for example, updating DECdfs (the Distributed File Service) entries to add access points.

DNS/BIND is the distributed name service for TCP/IP. It supports the storage of IP addresses and the use of node synonyms. Node synonyms allow for backward compatibility with older applications that cannot use long domain names. (DECnet-Plus also allows for node synonyms to provide backward compatibility with DECnet Phase IV node names.) DNS/BIND is needed if you want DECnet-Plus to run applications over TCP/IP. To use the DNS/BIND name service, DECnet-Plus requires one or more DNS/BIND servers in the network. DNS/BIND must be selected as one of the name services if you plan to use the DECnet over IP or OSI over TCP/IP features.

In general, use the Local namespace where possible, as it forces the network administrator to give more thought to the network node naming and addressing conventions used within an organization. In addition, once appropriate management procedures are in place, the simplicity of the Local namespace (not requiring configuration and management of servers as do DECdns and DNS/BIND) is much more preferable and can make fault finding much easier. All current network-related layered products (such as DECdfs) can operate in either a DECdns environment or in a local naming environment.

As noted, you can also use DECnet over IP, which uses the DNS/BIND name services as used in TCP/IP networks. This can greatly simplify consistent network-wide naming in a mixed-protocol environment. With DECnet over IP, end-to-end connectivity relies entirely on the underlying TCP/IP network configuration and infrastructure — the DECnet features of multiple paths and load balancing are no longer applicable; however, the availability features of TCP/IP (failSAFE IP) and OpenVMS (LAN failover) can be used instead. A DNS/BIND-less implementation is also possible by using the TCP/IP local HOSTS database to provide all the name resolution data.

One final note: the DECnet naming cache is non-volatile, meaning that it will survive reboots. If the naming information has changed, then you should make sure the naming cache is flushed to avoid the risk of stale cache entries and the confusion that follows. Flush the naming cache with the following command:

NCL FLUSH SESSION CONTROL NAMING CACHE ENTRY "*"

This will flush all cache entries. You can also use NCL to flush individual entries or groups of entries.

Availability and Reliability

DECnet and TCP/IP have very different approaches to re-routing on path failure. Other high-availability features related to LAN communications are now being built into the OpenVMS operating system.

DECnet node addressing is on a per-node basis and can thus provide both load balancing over all available data paths and automatic re-routing of data. These are handled by the end system with no external intervention and with minimal packet loss. With the appropriate changes to DECnet parameters, path failure can be detected quickly and failover can be achieved within a few seconds without disruption to the higher layer applications. This is best suited to a fully duplicated network infrastructure with separate multiple LANs.

In contrast to DECnet, TCP/IP addressing is on a per-interface basis. TCP/IP Services now provides failSAFE IP, which enables the IP address to be moved to a different physical interface when the primary interface fails or is no longer connected to the network infrastructure.

In addition, OpenVMS now provides mechanisms such as LAN failover so that all LAN protocols on a specific physical interface can move to an alternate physical interface when the primary interface fails or is no longer connected to the network infrastructure.

Node Names, Addresses, and Address Towers

Node names and synonyms help simplify the command line interface. For example, using node synonym XDVMS1, it is much easier to type the SET HOST command as SET HOST XDVMS1 rather than with the address as in SET HOST 10.240, or SET HOST 49::00-0A:AA-00-04-00-F0-28:21, or SET HOST IP$10.255.255.123. In this example, node XDVMS1 is running DECnet-Plus V7.3-1-ECO02 with Phase IV-compatible addressing enabled using OSI transport and the Local namespace. The true full node name is LOCAL:.XDVMS1.

The OpenVMS operating system uses the logical names SYS$NODE and SYS$NODE_FULLNAME for the Phase IV synonym and the full node name, respectively. The remote node (such as the target of a SET HOST command) implements the job logical names SYS$REM_NODE and SYS$REM_NODE_FULLNAME to provide the necessary information about the originating node for the incoming network connection.

The target node for a network request will perform a **back translation** of the inbound source address (from DECdns or the local namespace) to look up the corresponding name and will then cache it locally on the target node. This information is used to populate the SYS$REM_NODE and SYS$REM_NODE_FULLNAME job logical names on the target node so that software on the target node can easily determine where the inbound request originated. If the back translation fails, then the data provided in these logical names is simply the originator node address in DECnet, TCP/IP, or DECnet over IP format.

Nodes are registered in the DECdns or Local name databases using the DECNET_REGISTER tool. When a node is registered, the domain part (the part that precedes the ":.") is filled in with the appropriate domain name used by the relevant name service. The network administrator provides the name portion, the synonym (which defaults to the final part of the name portion after the "."), and the address tower information. If DECdns is used, then a node can **autoregister** its own address tower data based on its network adapter addresses, provided that DECdns access control has been configured correctly.

An **address tower set** stored in the namespace describes the protocols needed to establish a connection with a node. The address tower indicates which transport(s) to use to reach the destination node. The transports are either NSP or TP4, where NSP is the Phase IV compatible transport and TP4 is the OSI Class 4 transport. (Both transports can be used simultaneously but it is best to declare one only.) In addition, the address tower indicates which session control implementation is to be used. By default, SC3 is used for DECnet Phase V and SC2 for Phase IV. Finally, the address tower data contains the address field for the CLNS connection.

For example, the tower data for node XDVMS1 could appear as follows (using DECNET_REGISTER):

```
Address prefix = 49::
Fullname = LOCAL:.XDVMS1
Synonym = XDVMS1
Tower = SC2/NSP/CLNS=10.240,  Tower = SC3/TP4/CLNS=10.240
```

Notice that the tower data here contains two transport related entries for remote node XDVMS1 — one entry for OSI transport (SC3/TP4) and one for NSP (SC2/NSP). To connect to node XDVMS1 initiating only OSI transport connections (only available between Phase V nodes), simply delete the NSP (SC2/NSP) tower data entry for remote node XDVMS1 on the local node. To force initiation of only NSP connections (the only transport available on a Phase IV node), simply restrict the tower data to the NSP (SC2/NSP) entry by deleting the OSI transport (SC3/TP4) entry.

In general, do not attempt to use the OSI transport (SC3/TP4) for connecting your Phase V node to a Phase IV node (even with Phase IV-compatible address format). Otherwise, on attempting to connect to a Phase IV node, the initial connection attempt using the OSI transport will fail, and the connection attempt will have to be retried using the NSP transport.

For Phase V-to-Phase V communication, either transport can be used. Note that DECnet Phase V nodes will accept either transport for incoming connections using either SC2 or SC3. The address tower data is only used by the local node when initiating a connection to the remote node. In general, it is recommended to specify a single transport rather than both transports. This minimizes the timeout period if connection problems occur. If both transports are specified in the tower data, then both transports will be tried in succession, thus leading to a "double timeout," one per transport.

For detailed information on addresses for DECnet-Plus systems, including network service access points (NSAPs), service access points (SAPs), and OSI-style address formation, see the *DECnet-Plus Planning Guide*.

Phase IV Compatible Addressing

Phase V provides for a Phase IV-compatible address format. This is particularly relevant on CSMA/CD (Ethernet) LANs. The Phase IV address format uses the AA-00-04-00-*xx-xx* Ethernet MAC address, where the "*xx-xx*" portion is derived from the Phase IV area and node numbers. In contrast, the Phase V address format uses the hardware MAC address of the Ethernet adapter. This means that a pure Phase V node can use multiple adapters to connect to the same physical LAN while a Phase IV node cannot. DECnet Phase IV assumes one connection (station or network interface) per network segment, and multiple controllers are permitted only when there is no connection between the network segments, or when a DECnet router exists between the network segments. If more than one adapter attempts to start the DECnet protocols with the same (duplicate) MAC address, then subsequent adapters are prevented from coming on-line, although there can be timing issues with checking for duplicate MAC addresses, thus leading to unpredictable results.

This feature of Phase V provides the ability to load balance traffic over different adapters, which can be extremely useful especially for network reliability and throughput. It also permits the construction of bridged network configurations to provide increased availability of network paths. One major inconvenience is that existing LAN analyzers have to be reprogrammed to recognize a completely different set of Ethernet MAC addresses. Phase IV-compatible addressing still provides the ability to select the appropriate transport (NSP or OSI) for communication with other nodes, although only NSP is valid for communication with a pure Phase IV node.

Routing

With a mixed network of Phase IV and Phase V nodes and both NSP and OSI transports in use, the recommended routing mechanism is the Routing Vector Routing (RVR) algorithm as used by Phase IV Level 1 and Level 2 routers. DECnet-Plus host-based routing also uses the Routing Vector routing (RVR) algorithm. If the network has no Phase IV nodes and uses dedicated routers, then the recommended routing mechanism is Link State Routing (LSR) introduced with Phase V, which is faster to converge when the link topology changes. The main differences between the routing algorithms only become apparent in large networks with a large number of WAN links.

Prior to the release of DECnet-Plus V7.1 for OpenVMS V7.1, all routing functionality was provided by external dedicated routers (or OSI Intermediate Systems) such as DECnis routers. All Phase V implementations for OpenVMS were End Systems, analogous to the Phase IV End Node but with the ability to be Multi-Homed and thus they could drive several network paths in parallel simultaneously (giving features such as load balancing on dual-rail Ethernet).

DECnet-Plus V7.1 introduced host-based routing using the RVR mechanism. Host-based routing allows an OpenVMS system to operate as a DECnet-Plus intermediate system (IS). Host-based routing is analogous to the Phase IV Level 1 and Level 2 router functionality for OpenVMS systems that required a DECnet Extended Function (or Full-Function) license.

As already mentioned, host-based routing is useful for network configurations where data must be routed from a LAN to a wide area network (WAN) using an existing system rather than investing in a dedicated router. This will greatly benefit many network administrators who could not justify the purchase of additional hardware to provide routing functionality, although this was alleviated

somewhat by Phase V with the ability to create multi-homed End Systems. Multi-homed End Systems can have multiple simultaneously active network paths to different locations but cannot route between these paths. This feature can be extremely useful from a network security viewpoint when isolating systems and network segments from each other.

It is strongly recommended that all networks have at least one router node, even on a single LAN. Routing provides considerable additional functionality and the presence of a router assists with node unreachability detection and notification. Without a router node, unreachable status can only be determined by End System timeouts, which by default are set to expire after several minutes. The addition of routing functionality detects the loss of end system hello messages and then propagates routing updates, thus informing non-routers of node reachability changes. Routers also provide end systems with the necessary information to initiate a connection on the correct path without having to try paths and wait for timeouts.

DTSS – Time Synchronization Service

DTSS is the Distributed Time Synchronization Service. It is the topmost entity of the DECdts (DIGITAL Distributed Time Service) module. It provides a network-wide time synchronization mechanism that can keep clocks on all nodes accurately synchronized to within a small number of milliseconds. Not only can nodes be synchronized relative to each other but also relative to an external time source such as an IRIG-B signal or some other externally available clock. The basic concept behind DTSS is the introduction of a time specification that includes an inaccuracy value. The time stamp consists of the current time value with an associated inaccuracy. Without an external time provider, this inaccuracy value is infinite, indicating that all nodes are synchronized relative to each other and not with an external time reference.

An example time provider exists in SYS$EXAMPLES:. You can modify it to provide a time provider that simply reads the local clock and returns that time stamp with a small inaccuracy value, thus forcing the inaccuracy value to be non-infinite. Alternatively, code can be written to interface to an external dial-up or locally connected time reference source. Such devices are easily obtained, and many of them are simple radio clocks receiving a broadcast signal, for example from the Rugby clock in the UK. Coding a time provider requires use of the UTC time/date routines to generate the correct timestamp data structure with appropriate time values and inaccuracy values.

DTSS synchronizes the local node's clock with local (LAN connected with a non-DECnet layer 2 protocol) and global (DECnet connected over WAN or LAN) time servers. To synchronize, DTSS requires at least one time server to provide a time stamp with low inaccuracy. By default, DTSS is configured to communicate with a minimum of three servers (either local or global, or a mixture of both) to make an accurate synchronization. DTSS uses time stamp data from all three servers to make a balanced estimate of the actual time. If you have insufficient time servers available, then you will receive OPCOM messages indicating that too few servers were detected by the DTSS clerk.

To avoid the OPCOM messages, you can change the value of the SERVERS REQUIRED value to 1 in the DTSS startup NCL script (NET$DTSS_CLERK_STARTUP.NCL or NET$DTSS_SERVER_STARTUP.NCL), as shown in the example that follows, and have a single node as a DTSS global server, where *x* is the number of actual DTSS servers available on the local LAN or over WAN links and that are registered as Global Timeservers:

NCL SET DTSS SERVERS REQUIRED *x*

Register a node as a global server either by making an entry in the DECdns .DTSS_GlobalTimeServers directory (if DECdns is in use) or by making an entry in the SYS$MANAGER:DTSS$CONFIG.DAT file (if not using DECdns).

The selection of DTSS server or clerk software is made at the time of installation of the DECnet-Plus software. The DTSS startup procedure sets up the relevant time zone information and starts the

appropriate DTSS server or clerk software when the network is started at system boot. Time zone information is configured using NET$CONFIGURE in the usual manner.

It is also possible (again see SYS$EXAMPLES:) to use an NTP time server as a time source for a DTSS time provider process.

The simplest method of initiating time synchronization is to set the DTSS server's time manually from a watch, using an inaccuracy value of one second or so. When one time server has a low (non-infinite) inaccuracy component, the clocks on all other nodes will converge toward that server's time.

DTSS can also be disabled and prevented from starting on more recent versions of OpenVMS (V7.3-2 or later). You can prevent DTSS from starting at system boot by defining the NET$DISABLE_DTSS logical name in SYLOGICALS.COM. DTSS can be stopped on a running system by using the NCL DISABLE DTSS and NCL DELETE DTSS commands.

Associated Protocols – MOP and Remote Console

Maintenance Operation Protocol (MOP) is used to load executable code images into devices that boot from the local area network, such as DECserver terminal servers and OpenVMS cluster satellite nodes. As mentioned previously, as of OpenVMS V7.1 these load requests can be serviced by LANACP, DECnet Phase IV, or DECnet Phase V. The load host needs to have (1) the MOP service enabled and (2) a list of known devices and their corresponding MAC addresses, together with the load image file name and other associated data.

The Remote Console protocol is the mechanism used for establishing a remote console session on a network device, typically a DECserver.

With Phase V, use the following DCL SET HOST/MOP command to establish a remote console session. This command invokes the console carrier requester program and connects to the target device (*mop-client-name*) remote console interface.

SET HOST/MOP *mop-client-name*

Note that in contrast, Phase IV uses the NCP CONNECT NODE command to perform this function. In Phase IV, the target nodes need to be defined in the DECnet volatile database (usually loaded from the permanent database). This enables the MOP load request service mechanism to find the necessary load image and other node-specific data based on the Ethernet hardware MAC address information received from the device requesting a MOP load from the network.

The nodes booted in this manner do not necessarily run the DECnet protocol and thus do not really exist as DECnet nodes on the network. This has been a source of confusion for many system and network administrators. The term pseudo-node is more appropriate and the best practice is to allocate the DECnet Phase IV addresses to these pseudo-nodes so that they are easily distinguishable from the operational DECnet node addresses. For example, in a network where the various sites are split into areas 10, 11 and 12, the DECnet pseudo-node entries for MOP loaded devices could be in area 51, which does not physically exist in the chosen DECnet Phase IV addressing scheme. All MOP-loaded devices could safely be so configured across the entire network, assuming that there were less than 1023 such devices within the unused area 51. This would provide a clear separation of real DECnet nodes from MOP loaded devices not running DECnet, such as DECserver terminal servers.

Phase V introduced the more accurate term **MOP clients**. MOP clients do not need DECnet address information defined for them unless they are actually going to be running DECnet with the address information sent to the MOP client as part of the network boot process. This neatly distinguishes the terminal server type devices that need MOP to boot but do not run DECnet, from the cluster satellite type devices that run DECnet.

DECnet-Plus Installation Notes

Two DECnet license types are available: DVNETEND or DVNETEXT. Some of the installation options require the DVNETEXT license. The following are the license requirements for options discussed in this article:

- The DTSS server component does not require the DVNETEXT license.

- The DECdns server (re-introduced on OpenVMS Alpha V7.3-1) requires the DVNETEXT license.
- Host-based routing requires the DVNETEXT license. However, note that dedicated routers will always provide better functionality and greater performance for high traffic networks than host-based routers.

The OSI applications (VOTS, OSAK, FTAM) are installed separately from their kits located in subdirectories within the main kit directory structure.

The X.25 kit and WAN device drivers kit are separately licensed and installed products. The current version of X.25 is V1.6-ECO02. Note that X.25 V1.6-ECO02 is required for use on OpenVMS Alpha V7.3-2 (see the OpenVMS V7.3-2 release notes). X.25 V2.0 will support OpenVMS V8.2 Alpha and I64.

Changing the DECnet-Plus installation options (DTSS server, DNS server) after the initial installation requires any patch kits to be re-applied.

DECnet over IP Notes

Running DECnet over IP and OSI Applications over IP requires DECnet-Plus (or DECnet/OSI) and TCP/IP Services for OpenVMS V5.0 or later (or UCX V4.2-ECO04 or later).

Both OSI transport templates RFC1006 and RFC1006Plus must be configured in DECnet-Plus, and the PWIP driver must be configured in TCP/IP Services for OpenVMS. These provide the connection between DECnet and TCP/IP, allowing DECnet to use TCP/IP as a transport layer. This is implemented by the OSI Transport Templates for RFC1006 and RFC1006Plus (RFC1859), which map the data stream to TCP/IP port numbers 102 and 399. Note that it may be necessary to enable these TCP/IP port numbers on your firewall.

In addition, you must configure the DECnet naming service to use both the DNS/BIND (select "DOMAIN") and the Local or DECdns namespace options. This allows the TCP/IP naming to be resolved from the local TCP/IP HOSTS database or a TCP/IP DNS/BIND server. When using local naming for TCP/IP and DECnet, the order of name services is Local followed by DNS/BIND, and in either case the local node name (LOCAL:.*node-name* for DECnet, or the IP fully-qualified host name for TCP/IP) is the name of the relevant naming service. This allows DECnet to use the local naming file and TCP/IP to use the local HOSTS file. If you are using the local TCP/IP HOSTS database, then specify the TCP/IP DNS/BIND server as the local TCP/IP node.

DECnet will first attempt to establish a connection by looking up the name in the Local or DECdns namespace. It then resolves that name into the DECnet address. If DECnet over IP is enabled (RFC1006 & RFC1006Plus OSI transport templates, PWIP driver enabled and the DOMAIN name added to the local name path search list), then DECnet queries the local TCP/IP BIND resolver, which in turn looks up the local TCP/IP HOSTS database or queries the local TCP/IP BIND server to resolve the name into a TCP/IP address. This is configured by (re)running NET$CONFIGURE in ADVANCED mode to change the naming services and the OSI transport NCL scripts once TCP/IP Services for OpenVMS has been installed, configured (using TCPIP$CONFIG), and started with the PWIP driver enabled. When configuring the OSI transport, answer "Yes" to the "Do you want to run OSI applications over IP" and "Do you want to run DECnet over IP" questions.

Useful Commands:

- To list the BG devices, corresponding ports, and so forth:

  TCPIP SHOW DEVICE

  This command should display the RFC1006 and RFC1006Plus ports (102 and 399, respectively). DECnet over IP needs the RFC1006Plus port for allowing the IP data stream to be handed to the RFC1006Plus OSI transport template in DECnet. For full details of the extensions to RFC1006, see RFC1859.

- List the active OSI transport templates:

  NCL SHOW OSI TRANSPORT TEMPLATE * ALL

The display should confirm the presence of the RFC1006 and RFC1006Plus OSI transport templates. The NCL commands to create and start those OSI transport templates are in the NET$OSI_TRANSPORT_STARTUP.NCL, which is created by NET$CONFIGURE.

Current Versions of DNA Phase V Implementations

As of the time of writing (February 2005), several Phase V implementations exist, depending on the version of OpenVMS and the type of hardware in use:

- DECnet/OSI V6.3 with ECO16 on OpenVMS VAX V6.2
- DECnet/OSI V6.3 with ECO16 on OpenVMS Alpha V6.2-1H3
- DECnet-Plus V7.3 with MUP01 and ECO04 on OpenVMS VAX V7.3
- DECnet-Plus V7.3-1 with MUP01 and ECO03 on OpenVMS Alpha V7.3-1
- DECnet-Plus V7.3-2 with MUP01 and ECO01 on OpenVMS Alpha V7.3-2
- DECnet-Plus V8.2 with ECO01 on OpenVMS Alpha V8.2
- DECnet-Plus V8.2 with ECO01 on OpenVMS I64 V8.2

Make sure all relevant patches have been applied to the base operating system, especially the LAN driver patches for V7.3, V7.3-1 and V7.3-2.

For more information

Further information is available at:

- http://www.hp.com/go/openvms/doc - OpenVMS Documentation
- http://www.hp.com/go/openvms/wizard - Ask the Wizard
- http://www.research.digital.com/wrl/DECarchives/DTJ
- http://ftp.digital.com/pub/DEC/DECnet/PhaseIV/ — DNA Phase IV specifications
- http://rfc.net then searching for “decnet”
- http://standards.ieee.org/getieee802/802.3.html — IEEE802.3 (Ethernet) specifications
- http://linux-decnet.sourceforge.net

You can contact the author directly by e-mail at vmstj@xdelta.co.uk.

OpenVMS Technical Journal V5

Delivering Web Access to OpenVMS

Delivering Web Access to OpenVMS

Tom Bice, Manager of Integration Strategy for WRQ

Within the HP community, IT decision-makers face many critical technology issues. For Lexington, S.C.-based multi-million dollar carrier company, Southeastern Freight Lines (SEFL), modernizing the assets in its OpenVMS system using host integration software turned an IT challenge into a unique business opportunity.

To power its IT infrastructure, SEFL built a portal to improve the management of its high-volume, transaction-based shipping operations. However, the majority of its critical claims, billing and shipping trip information was locked away on their OpenVMS data system. SEFL needed to retain use of these critical OpenVMS applications but also provide transparent Web access to the data via the portal.

Host integration software helps companies like SEFL integrate their mission-critical host applications and data sources into modern applications such as portals, web services and other web-based applications,

SEFL relies on OpenVMS to provide stability and security and ensure zero downtime to run its operations. It has been using OpenVMS for more than 20 years to support mission critical business processes and house the majority of its vital corporate assets. These legacy applications are complex – both for users and developers. Therefore, taking the time and expense to re-engineer them is not always an option. What was once cutting edge technology had become a critical legacy asset, and SEFL had to decide how to retain the decades of critical business data residing there.

SEFL, which specializes in hauling carpet and carpet-related products through its fleet of about 2,600 tractors and 7,500 trailers, makes 25,000 shipments daily. Each stop along the way is recorded electronically, resulting in millions of transactions to track the cost and efficiency of the SEFL shipping systems. To do this, SEFL uses what is perhaps the company's most critical application, All-IN-1, a mainframe email and POP system. SEFL relies on this OpenVMS character-based application for tracking billing, claims and trip tickets – an integral part of its business. Because ALL-IN-1 will not be ported to a modern-day computer platform, SEFL had to find an appropriate solution to reuse the data and logic in a web environment that users and developers were more familiar with. While the team quickly selected products like Microsoft Exchange to automate the company's email needs, the integration of the rest of the company's decades worth of business logic – 1,600 ALL-IN-1 programs running on Cobol – proved to be more difficult.

Dave Robinson, vice president for MIS, and his team discussed creating a graphical user interface (GUI) that would modernize these thousands of programs and make them web-accessible, instead of accessible only from the mainframe. This would rapidly address new business initiatives, while reducing costs. The browser-based application would be presented on screens in a GUI format and that information would be sent to the OpenVMS system in the form of business logic that the new system could read. To keep all of its business files the same without undergoing a major conversion, SEFL had to find a product that would allow it to program all of this data in an OpenVMS environment.

"OpenVMS is a scalable, dependable operating system that we really love," Robinson said. "We did not want to take our business off OpenVMS, but we also did not want to lose all of the data that's key to our business."

With millions of dollars invested in OpenVMS, including 40 MIS employees who are experts in the operating system, Robinson could not afford to eliminate these assets. He looked for solutions that would allow SEFL to get more out of its existing OpenVMS investment in people, process and technology. In IT, where budgets and headcount determine the company's ability to meet increasing demands put on them to become more agile, decreasing the project timeline, reacting quickly to changing business requirements, and ultimately creating an IT environment that is flexible enough to be proactive were essential.

Therefore, he sought a solution that would modernize the company's existing legacy assets residing on OpenVMS; support the organization's file structure; and address the organization's integration, migration and web access needs. He knew going into the project that only a handful of products

would address both business and IT needs and extend his current investment in OpenVMS legacy assets. He evaluated three, including WRQ Verastream host integration server. A big believer in prototypes, Robinson spent six weeks evaluating one product option and then evaluated the WRQ solution, which successfully built a prototype in only a week.

"We had numerous and complex demands, ranging from very high performance requirements to the fundamental need to retain our OpenVMS-based system. Any amount of downtime is unacceptable, which is why we run OpenVMS in the first place. We needed a vendor that would stand behind their software, uptime and scalability," Robinson said. "WRQ demonstrated, met, and exceeded all of our requirements."

With WRQ's host integration solution, SEFL was able to modernize its infrastructure and applications, while maximizing the value of its OpenVMS system.

As a result of using WRQ Verastream, SEFL would web-enable and integrate all of the company's mission-critical business operations in only a matter of months instead of years – completely replacing the All-IN-1 system.

WRQ first tackled the need to provide web access and automate SEFL's most important application: an OpenVMS character-based application used for billing, claims and trip tickets. WRQ completed a custom configuration and deployment of WRQ Reflection for the Web terminal emulation in less than eight weeks to handle this requirement. Now, SEFL's 6,000 employees and all of its customers have transparent access to OpenVMS applications, while automatically selecting and loading appropriate documents from multiple business systems.

WRQ experts then proceeded with SEFL's portal development, using WRQ Verastream at the backend. Through the use of auto-generated customizable portlets, Verastream allows SEFL users to go to a centralized, web-based location for fast and easy access to legacy application information. Because WRQ Verastream has the ability to integrate interchangeable components, WRQ developed the portal without replicating data or rewriting code from the underlying OpenVMS system. All data – old and new – would reside in the new system WRQ created.

Converting its external web development from an ASP to a Java environment using Verastream meant that SEFL's Cobol developers had to learn Java, a completely new programming language. The use of Java required a completely different approach, but SEFL's developers quickly adopted this new programming language largely for the productivity improvements that it brings. A web and Java interface allows SEFL to support all of its technology and application development without making major changes. Because Verastream is a complete integration solution, SEFL could easily access data, integrate with user interfaces, or integrate on an application level – giving the company a "one-stop shop" for its integration needs.

Now because employees can access applications on the Web, they are also quickly becoming more productive. Previously, each user would have to scan hundreds of screens before finding the one – out of 2,800 menu options – that ran the application they were looking for. Part of the conversion entailed customizing each user's menu. With the web-based interface, only the options specific to each user's tasks are displayed; for most users, a 2,800-menu system is now just a 50-menu system. Throughout the migration, users can log on to their ALL-IN-1 applications as usual and see the interface they have been using for years. As each department's migration is complete, users will see an easy-to-use, web-enabled Verastream interface. To make the transition even smoother, applications are converted on a set schedule.

Additionally, the transition to a GUI environment from a mainframe terminal meant SEFL employees found it much more intuitive because of the web interface. SEFL could now query host information as if it were a database. As a result, SEFL enhanced speed of customer service and reduced IT costs through improved efficiency.

In fact, Robinson recalls many users proclaiming "It's about time we get out of the Dark Age," when first hearing about the transition. Cobol and ALL-IN-1 were state-of-the-art 20 years ago, but SEFL continued to develop business applications ever since, using these processes that quickly became antiquated. Now, novice users that could barely perform simple word processing functions can send spreadsheets to customers. In addition, SEFL users can access applications from their home computer's web browser. This was an added advantage during the recent hurricane season when employees could check in and use applications from home rather than brave the storms to get to the office.

All SEFL email is now converted to Microsoft Exchange, and in 2005, 100 percent of ALL-IN-1 programs will be eliminated and converted to Verastream.

The catalyst for this project was the forthcoming elimination of the ALL-IN-1 program. But had it not been for Verastream, Robinson notes that his organization would be faced with radical technology transformations such as changing the operating system, changing computer systems, and rewriting the company's entire Cobol environment. A homegrown solution just wasn't an option.

"It would be mind boggling if we couldn't salvage all that we could, namely the file system," he says. "Without Verastream, we would be looking at a 20 man-year project. It would be like bulldozing a building and starting from scratch."

For more information

For more information, you may contact Tom Bice at tomb@wrq.com, or you can visit WRQ at www.wrq.com.