Application Design in a VAXcluster System

By William E. Snaman, Jr.

Abstract

VAXcluster systems provide a flexible way to configure a computing system that can survive the failure of any component. In addition, these systems can grow with an organization and can be serviced without disruption to applications. These features make VAXcluster systems an ideal base for developing high-availability applications such as transaction processing systems, servers for network client-server applications, and data sharing applications. Understanding the basic design of VAXcluster systems and the possible configuration options can help application designers take advantage of the availability and growth characteristics of these systems.

Many organizations depend on near constant access to data and computing resources; interruption of these services results in the interruption of primary business functions. VAXcluster systems provide solutions to these data availability and growth problems that modern organizations face.[1]

This paper begins with an overview of VAXcluster systems and application design in such systems and proceeds with a detailed discussion of VAXcluster design and implementation. The paper then focuses on how this information affects the design of applications that take advantage of the availability and growth characteristics of a VAXcluster system.

Overview of VAXcluster Systems

VAXcluster systems are loosely coupled multiprocessor configurations that allow the system designer to configure redundant hardware that can survive most types of equipment failures. These systems provide a way to add new processors and storage resources as required by

In addition, growing organizations face the need to increase the amount of computing power available to them over an extended period of time. VAXcluster the organization. This feature eliminates the need either to buy nonessential equipment initially or to experience painful upgrades and application conversions as the systems are outgrown.

The VMS operating system, which runs on each processor node in a VAXcluster system, provides a high level of transparent data sharing and independent failure characteristics. The processors interact to form a cooperating distributed operating system. In this system, all disks and their stored files are accessible from any processor as if those files were connected to a single processor. Files can be shared transparently at the record level by application software.

To provide the features of a VAXcluster system, the VMS operating system was enhanced to facilitate this data sharing and the dynamic adjustment to changes in the underlying hardware configuration. These enhancements make it possible to dynamically add multiple processors, storage controllers, disks, and tapes to a VAXcluster system configuration. Thus, an organization can purchase a small system initially and expand as needed. The addition of computing and storage resources to the existing configuration requires no software modifications or

Application Design in a VAXcluster Environment

Application design in a VAXcluster environment involves making some basic choices. These choices concern the type of application to be designed and the method used to synchronize the events that occur during the execution of the application. The designer must also consider application communication within a VAXcluster system. A discussion of these issues follows.

General Choices for Application Design

This section briefly describes the general choices available to application designers in the areas of client-server computing and data access.

Client-server Computing. The VAXcluster environment provides a fine base for client-server computing. Application designers can construct server applications that run on each node and accept requests from clients running on nodes in the VAXcluster system or elsewhere in a wider network.

If the node running a server application fails, the clients of that server can switch to another server running application conversions and can be accomplished without shutting down the system. The ability to use redundant devices virtually eliminates single points of failure.

on a surviving node. The new server can access the same data on disk or tape that was being accessed by the server that failed. In addition, the redundancy offered by the

VMS Volume Shadowing Phase II software eliminates data unavailability in the event of a disk controller or media failure.[2] The system is thus very available from the perspective of the client applications.

Access to Storage Devices. Many application design questions involve how to best access the data stored on disk. One major advantage of the VAXcluster system design is that disk storage devices can be accessed from all nodes in an identical manner. The application designer can choose whether the access is simultaneous from multiple nodes or from one node at a time. Consequently, applications can be designed using either partitioned data access or shared data access.

Using a partitioned data model, the application designer can construct an application that limits data access to a single node or subset of the nodes. The application runs as a server on a single node and accepts requests from other nodes in the cluster and in the network. And because the application runs on a single node, there is no need for synchronization of local buffer caches and can aggregate larger amounts of data for write operations, thus minimizing I/O activity.

An application that uses partitioned data access lends itself to many types of high-performance database and transaction processing environments. VAXcluster systems provide such an application with the advantage of having a storage medium that is available to all nodes even when they are not actively accessing the data files. Thus, if the server node fails, another server running on a surviving node can assume the work and be able to access the same files. For this type of application design, VAXcluster systems offer the performance advantages of a partitioned data model without the problems associated with the failure of a single server.

Using a shared data model, the application designer can create an application that runs simultaneously on multiple VAXcluster nodes, which naturally share data in a file. This type of application can prevent the bottlenecks associated with a single server and take advantage of opportunities for parallelism on need to synchronize data access with other nodes. Eliminating this source of communication latencies can improve performance in many applications. Also, if synchronization is not required, the designer can make the best use

multiple processors. The VAX RMS software can transparently share files between multiple nodes in a VAXcluster system. Also, Digital's database products, such as Rdb/VMS and VAX DBMS software, provide the same data-sharing capabilities. Servers running on multiple nodes of a VAXcluster system can accept requests from clients in the network and access the same files or databases. Because there are multiple servers, the application continues to function in the event that a single server node fails.

Application Synchronization Methods

The application designer must also consider how to synchronize events that take place on multiple nodes of a VAXcluster system. Two main methods can be used to accomplish this: the VMS lock manager and the DECdtm services that provide VMS transaction processing support.

VMS Lock Manager. The VMS lock manager provides services that are flexible enough to be used by cooperating processes for mutual exclusion, synchronization, and event notification.[3] An application uses these services either directly or indirectly through components of the system such as the VAX RMS software.

DECdtm Services. The VMS operating system provides a set of services to facilitate transaction The services use a two-phase commit protocol. A transaction may span multiple nodes of a cluster or network. The support provided allows multiple resource managers, such as the VAX DBMS, Rdb/VMS, and VAX RMS software products, to be combined in a single transaction. The DECdtm transaction processing services take advantage of the guarantees against partitioning, the distributed lock manager, and the data availability features, all provided by VAXcluster systems.

VAXcluster and Networkwide Communication Services

Application communication between different processors in a VAXcluster system is generally accomplished using DECnet task-to-task communication services or other networking software such as the transmission control protocol (TCP) and the internet protocol (IP). Client-server applications or peer-to-peer applications are easy to develop with these services. The services allow processes to locate or start remote servers and then to exchange messages.

Since the individual nodes of a VAXcluster system exist as separate entities in a wider processing.[4] These DECdtm services enable the application designer to implement atomic transactions either directly or indirectly.

communication network, applications communication inside a VAXcluster system can rely on general network interfaces. Thus, no special-purpose communication services were developed. Applications are simpler to design when they can communicate within the cluster in the same manner in which they communicate with nodes located outside the VAXcluster system.

A DECnet feature known as cluster alias provides a collective name for the nodes in a VAXcluster system. Application software can connect to a node in the cluster using the cluster alias name rather than a specific node name. This feature frees the application from keeping track of individual nodes in the VAXcluster system and results in design simplification and configuration flexibility.

VAXcluster Design and Implementation Details

To understand how the design and implementation of a VAXcluster system affects application design, one must be familiar with the basic architecture of such a system, as shown in Figure 1. This section describes the layers, which range from the communication mechanisms to the users of the system.

Port Layer

The port layer consists of Several of the ports utilize multiple physical communication paths, which appear as a single logical path to the VAXcluster software. This redundancy provides better communication throughput and higher availability. If multiple logical paths exist between a pair of nodes, the VAXcluster software generally selects one for active use and relies on the remaining paths for backup in the event of failure.

The port layer can contain any of the following interconnects:

o Computer Interconnect (CI) bus

o Ethernet

o Fiber distributed data interface (FDDI)

o Digital Storage Systems Interconnect (DSSI) bus

Each bus is accessed by a port (also called an adapter) that connects to the processor node. For example the CI bus is accessed by way of a CI port. The various buses provide a wide spectrum of choices in terms of wire and adapter capacity, number of nodes that can be attached, distance between nodes, and cost.[5]

the lowest levels of the architecture, including a choice of communication ports and physical paths (buses). The VAXcluster software requires at least one logical communication pathway between each pair of processor nodes in the VAXcluster system.

The CI bus was designed for access to storage and for reliable host-to-host communications. Each CI port connects to two redundant, high-speed physical paths. The CI port dynamically selects one of the two paths for each transmitted message.

Messages are received on either path. Thus, two nodes can communicate on one path at the same time that two other nodes communicate on the other. If one physical path fails, the port simply uses the remaining path. The existence of the two physical paths is hidden from the software that uses the CI port services. From the standpoint of the cluster software, each port represents a single logical path to a remote node. Multiple CI ports can be used to provide multiple logical paths between pairs of nodes. An automatic load-sharing feature distributes the load between pairs of ports.

The DSSI bus was primarily designed for access to disk and tape storage. However, the bus has proven an excellent way to connect small numbers of processors using the VAXcluster protocols. Each DSSI port connects to a single high-speed physical path. As in the case of the CI bus, several DSSI ports may be connected to a node to provide redundant paths. (Note that the KFQSA DSSI port is for storage access only and provides no general communication service between nodes.)

Ethernet and FDDI are open local area networks, generally shared by a wide variety of consumers. Consequently, the VAXcluster software was designed to use the Ethernet and FDDI ports and buses simultaneously with the DECnet or TCP/IP protocols. This is accomplished by allowing the Ethernet data link software to control the hardware port. This software provides a multiplexing function such that the cluster protocols are simply another user of a shared hardware resource.

Each Ethernet and FDDI port connects to a single physical path. There may be more than one port on each processor node. This means that there may be many separate paths between any pair of nodes when multiple ports are used. The port driver software combines the multiple Ethernet and FDDI paths into a single logical path between any pair of nodes. The load is automatically distributed among the various possible physical paths by an algorithm that chooses the best path in terms of adapter capacity and path latency.[6]

System Communications Services Layer

The system communications services (SCS) layer of the VAXcluster architecture is implemented in a combination of hardware and software or software only, depending upon the type of port. The SCS layer manages a logical path between each pair of nodes in the VAXcluster system. This logical path consists of a virtual circuit (VC) between each pair of SCS ports and a set of SCS connections that are multiplexed on that virtual circuit. The SCS provides basic connection management and communication services in the form of datagrams, messages, and block transfers over each logical path.

The datagram is a best-effort delivery service which offers no guarantees regarding loss, duplication, or ordering of datagrams packets. This service requires no connection between the communicating nodes. In general, the VAXcluster software makes minimal use of the datagram service.

The message and block transfer services take place over an SCS connection. Consumers of SCS services communicate with their counterparts on remote nodes using these connections. Multiple connections are multiplexed on the logical path provided between each pair of nodes in the VAXcluster system.

The message service is reliable and guarantees that there will be no loss, duplication, or permutation of message sequence on a given connection. The connection will break accomplishes the block transfer, thus freeing the host processor to perform other tasks. Some DSSI ports use hardware to copy data and others rely on software to perform this function. Depending on the exact model of an Ethernet or FDDI port, the port software, rather than the hardware, moves the data.

System Applications

The next higher layer in the VAXcluster architecture consists of multiple system applications (SYSAPs). These applications provide, for example, access to disks and tapes and cluster membership control. The following sections describe some SYSAPs.

Connection Manager. The connection manager serves three major functions. First, the connection manager knows which processor nodes are active members of the VAXcluster system and which are not. This is accomplished through a concept of cluster "membership." Nodes are explicitly added to and removed from the active set of nodes by a distributed software algorithm. In a VAXcluster system, every processor node must have an open SCS connection to all other processor nodes. Once rather than allow the consumer of the service to perceive such errors.

The block transfer service provides a way to transfer quantities of data directly from the memory of one node to that of another. For CI ports, the port hardware

a booting node establishes connections to all other nodes currently in the VAXcluster system, this node can request admission to the system. When one node is no longer able to communicate with another node, one of the two nodes must be removed from the VAXcluster system.

In a VAXcluster system, all nodes have a consistent view of the cluster membership in the presence of permanent and temporary communication failures. This consistency is accomplished by using a two-phase commit protocol to form the cluster, add new nodes, and remove failed nodes.

The second function provided by the connection manager is an extension of the SCS message service. This extension guarantees that the service will (1) deliver a message to a remote node or (2) remove either the sending node or the receiving node from the cluster. The strong notion of cluster membership provided by the connection manager makes this guarantee possible. The service attempts to deliver the queued messages to remote nodes. If a connection breaks, the service attempts to reestablish communication to the remote node and resend the message. After a period of time specified by the system manager, the service declares the connection irrevocably broken and removes either the sending or the receiving node from the VAXcluster membership.

This message service allows users to construct efficient protocols that do not require acknowledgment of messages. The service proved to be a very powerful tool in the design of the VMS lock manager. The delivery guarantees inherent in the service minimize the number of messages required to perform any given locking function, resulting in a corresponding increase in performance. The ability to hide failures by updating cluster membership further simplified the lock manager design and increased performance; this capability enabled the removal of logic used to handle changes in VAXcluster configurations and communication errors from all main lock manager code paths.

The third function of the connection manager is to prevent partitioning of the possible cluster members. Partitioning of a system exists when separate processing elements function independently. If a system allows data sharing, completely independent processing can result in uncoordinated access to shared resources and lead to data corruption.

In a VAXcluster system, processors communicate

Thus, the service hides all temporary communication failures from its client.

and coordinate access to resources by means of a voting algorithm. The system manager assigns a number of votes to each processor node based on the importance of that node. The system manager also informs each node of the total number of possible votes. The algorithm requires that more than half of these votes be present in a VAXcluster system for nodes to function. When the sum of all votes contributed by the members of a VAXcluster system falls below this quorum, the VMS software blocks I/O to mounted devices and prevents the scheduling of processes. As nodes join the cluster, votes are added. Activity resumes once a quorum is reached.

In practice, the connection manager uses two measurements of the number of votes: static and dynamic. The static count of votes is the globally agreed on number of votes contributed by cluster members. This count is created ignoring the state of connections between nodes. The value of the static quorum changes only at the completion of two-phase commit operations, which accomplish a user-requested quorum adjustment in addition to performing the other activities mentioned earlier in this Connection Manager section.

Each node independently maintains the dynamic blockage of process and I/O activity.

To provide configurations with a small number of nodes, e.g., two nodes, the concept of a quorum disk was invented. The system manager assigns a disk to contribute votes to the cluster. A node must be able to access a file on the disk in order to include the votes assigned to that disk in the node's own total. Consequently, a special algorithm is used to access the file. This algorithm ensures that two unrelated nodes cannot both count the quorum disk votes. Doing so could result in partitioned operation.

Mass Storage Control Protocol Server. The Mass Storage Control Protocol (MSCP) server allows disks that are attached to one or more VAX processors to be accessed by other processors in the VAXcluster system. Thus, a VAXcluster processor may emulate a multihost disk controller by accepting and processing I/O requests from other nodes and accessing the disk indicated by the request. The server can process multiple commands simultaneously and also performs fragmentation of count. This count represents the sum of all votes contributed by VAXcluster members with which the tallying node has a functional connection. Changes in the dynamic quorum, and not the static quorum, initiate the

commands if there is not enough system buffer space to accommodate the entire amount of data at one time.

Hierarchical Storage Controllers, Local Controllers, and RF-series Integrated Storage Elements. Hierarchical storage controller (HSC) servers are specialized devices that perform MSCP serving of RA-series disk drives and TA-series tape drives in a VAXcluster system. HSC servers connect directly to the CI bus. In addition to providing the host with access to the storage media, HSC servers accomplish performance optimizations such as seek-ordering and request fragmentation based on real-time head position information. The local disk controllers attached to the RA- and TA-series storage devices perform the same function for a single host processor. The RF-series integrated storage elements (ISEs) attach to a DSSI bus. Each of these disk storage devices performs its own command queuing and optimization without using a dedicated controller.

Disk Class Driver. The disk class driver allows access to disks served by an MSCP server, an HSC controller, a local Digital storage architecture (DSA) controller, or attached to a DSSI bus. This driver provides a command queuing function that allows a disk controller to have restarts commands as needed.

VAXcluster systems can be configured so that all disks are accessed by way of redundant paths for increased availability. The way in which this is accomplished depends on the type of disk and the disk controller.

RF-series disks contain integrated controllers that connect to a single DSSI storage bus. This bus can be accessed by up to two VAX processors. Each VAX processor can then serve the disks to all other nodes in the VAXcluster system. Thus, two paths are provided to each disk.

RA-series disks connect to up to two storage controllers. These controllers can be either (1) local adapters attached directly to a single processor node or (2) HSC controllers located on the CI bus. Disks connected to local adapters can be served to other nodes of the VAXcluster system. Disks located on an HSC controller can be directly accessed by processors that are not on that bus. Thus, the use of multiple controllers when combined with disk serving provides at least two paths to a disk from every node in the VAXcluster system.

multiple outstanding commands which can be used to provide seek, rotation, and other performance optimizations. To handle temporary communication interruptions, the driver

Since many paths exist to gain access to a disk, the disk class driver chooses which path to use when a disk is initially mounted by a node. If the path to the disk becomes inoperative, the disk class driver locates another path and begins to use it. Server load and type of path, i.e., local or remote, are considered when selecting the new path. This reconfiguration is totally transparent to the end user of the disk I/O service.

Tape Class Driver. The tape class driver performs functions in a VAXcluster system similar to those of the disk class driver by providing access to tapes located on HSC controllers, local controllers, and DSSI buses.

VMS Components Layered on Top of SYSAPs

The SYSAPs provide basic services that other VMS components use to provide a wide range of VAXcluster features.

Volume Shadowing. The volume shadowing product allows multiple disks to be utilized as a single, highly available disk. Volume shadowing provides transparent access to the data in the event of disk media or controller failures, media degradation, and communication failures.[2] The shadowing layer works in conjunction with the disk class driver to accomplish this task. With

Lock Manager. The VMS lock manager is a system service that provides a distributed synchronization function used by many components of the VMS operating system, including volume shadowing, the file system, VAX RMS software, and the batch /print system. Application programs can also use the lock manager directly.

The lock manager provides a name space that is truly clusterwide. Cooperating processes can request locks on a specific resource name. The lock manager either grants or denies these requests. Processes can also queue requests. The lock manager services allow processes to coordinate the means of access to physical resources or simply provide a communication pathway between processes. Processes can use the service for such tasks as mutual exclusion, event notification, and server failure detection.[2,7] The lock manager uses the communication service provided by the connection manager to minimize the message count for a given operation and to simplify the design by eliminating the need to consider changes in cluster membership from all main paths of operation.

the advent of VMS Volume Shadowing Phase II, disk shadowing is extended to many new configurations.

Process Control Services. The VMS process control system services take advantage of VAXcluster systems. Applications can use these services to alter process states on remote nodes and to collect information about those processes. In the future, it is likely that other services will be extended to make optimal use of VAXcluster capabilities.

File System. The VMS file system (XQP) allows disk devices to be accessed by multiple nodes in a VAXcluster system. The file system uses the lock manager to coordinate disk space allocation, buffer caches, modification of file headers, and changes to the directory structure.[8]

Record Management Services. The VAX RMS software allows the sharing of file data by processes running on the same or multiple nodes. The software uses the lock manager to coordinate access to files, to record data within files, and to global buffers.

Batch/Print System. The batch/print system allows users to submit batch or print jobs on one node and run them on another. This system provides a form of load distribution, i.e., generic batch queues can feed executor queues on each node. Jobs running on a failed node can be restarted automatically on another node in the VAXcluster system.

An Application Constructed Using VAXcluster Mechanisms

down into various phases such as fetch sources, compile, and link. The phases must execute in a given order but are otherwise independent. Each phase can be restarted from the beginning if there is an error. Each major component of the VMS operating system is processed separately during each of the phases. All sources reside on a shared disk to which all nodes of the VAXcluster system have access; the output disk is shared by all nodes also. A master data file describes the phases and the components. For a given phase, the actions required for each component are fed into a generic batch queue. This queue feeds the jobs into work queues on multiple nodes, resulting in the execution of many jobs in parallel. When all jobs of a phase have completed, the next phase starts. If a node fails during the execution of a job, that job is restarted automatically on another node either from the beginning or from a checkpoint in the job. This use of shared disks and batch queues provides great parallelism and reliability in the VMS build process.

The Impact of VAXcluster Design and Implementation on

The VMS software build process is an example of how these mechanisms can be used to benefit application design. The VMS software build is broken

Applications

This section discusses how multiple communication paths, membership changes, disk location and availability, controller selection, disk and tape path changes, and disk failure impact application design.

Multiple Communication Paths

VAXcluster software components are able to take advantage of multiple communication paths between nodes. For greatest availability, there should be at least two physical paths between each pair of nodes in a VAXcluster system.[6]

Membership Changes

VAXcluster membership changes involve several distinct phases with slight variations depending upon whether a node is being added or removed. Adding a node to a VAXcluster system is the simplest case because it involves reconfiguration. There is a further simplification in that nodes are only added one at a time. A booting node petitions a member of an existing cluster for membership. This member then describes the booting node to all other member nodes and vice versa. In this way, it is determined that the booting node is in communication with all members of the cluster. The connection manager then adds the new node to the cluster using a two-phase commit protocol to ensure a

Removing a node is more complicated because both failure detection and reconfiguration must take place. In many cases, there may be multiple simultaneous failures of nodes and communication paths. The view of what nodes are members and which paths are functional may be very different from each node. Additionally, new failures may occur while the cluster is being reconfigured.

The initial phase involves the detection of a node failure. A node may cease processing, but other cluster members may not be aware of this fact. The communication components generally exchange messages periodically to determine whether other nodes are functioning. The first indication of a failure may be the lack of response to these messages. However, a minimum period of time must elapse before the connection is declared inoperative. This set delay prevents breaking connections when the network or remote system is unable to respond due to a heavy load. Once the communication failure is detected, the connection manager is notified by the SCS communication layer. The connection manager consistent membership view from all nodes.

attempts to restore the connection for a time interval defined by the system manager using a system control parameter known as RECNXINTERVAL. Once this interval has expired, the connection and hence the remote node is declared inoperative. The connection manager then begins a reconfiguration.

Multiple nodes may attempt the reconfiguration at the same time. A distributed election algorithm is used to select a node to propose the new configuration. The elected node proposes to all other nodes that it can communicate with a new cluster configuration that consists of the "best" set of nodes that have connections between each other. "Best" is determined by the greatest number of possible votes. If multiple configurations are possible with the same number of votes, the configuration with the most nodes is selected.

Any node that receives the proposal and can describe a better cluster rejects the proposal. The proposing node then withdraws the proposal and the election process begins again. This cycle continues until all nodes accept the proposal. The cluster membership is then altered using a two-phase commit protocol, removing nodes as required.

Even when one considers the worst case of a continual failure situation, convergence on a solution is guaranteed because the connection manager does not add nodes as a result of nodes rebooting or because failed connections are restored. Conditions can only get worse, i.e., simpler, until failures cease to happen long enough for the reconfiguration to complete.

However, this worst-case condition is atypical; most reconfigurations are very simple. A node that is removed, as a result of a planned shutdown or because it fails, attempts to send a "last gasp" datagram to all VAXcluster members. This datagram indicates that the node is about to cease functioning. The delay present during the failure detection phase is bypassed completely, and the connection manager configures a new VAXcluster system in considerably less than one second.

Normally, the impact on an application of a node joining a VAXcluster system is minimal. For some configurations, there is no blockage of locking. In other cases, the distributed directory portion of the lock database must be rebuilt. This process may block locking for up to a small number of seconds, depending on the number of nodes, number of directory entries, and type of communication buses in not add new nodes during

a reconfiguration and connections that fail are never used again. Thus, conditions cannot oscillate between good and bad during the reconfiguration because

use.

Application delays can result when an improperly dismounted disk is mounted by a booting node. Failure to properly dismount the disk, e.g., because of a node failure, results in the temporary loss of some preallocated resources such as disk blocks and header blocks. An application can recover these resources when the disk is mounted, but the I/O is blocked to the disk during the mounting operation. This I/O blocking has a potentially detrimental impact on applications that are attempting to allocate space on the disk. The answer to this problem is to mount disks so that the recovery of the preallocated resources is deferred. For all disks except the system disk, disk mounting is accomplished with the MOUNT /NOREBUILD command. Because a system disk is implicitly mounting during a system boot, the system parameter ACP_REBLDSYSD must be set to the value 0 to defer rebuilds. The application can recover the resources at a more opportune time by issuing a SET VOLUME /REBUILD command.

The impact on a VAXcluster system of removing a node varies depending on what resources the application needs. During the failure detection phase, messages to a failed node may be queued pending discovery that there actually is a failure. If the application lock manager may experience a delay, but as long as there are sufficient votes present in the cluster to constitute a quorum, the I/O is not blocked during the reconfiguration. If the number of votes drops below a quorum, I/O and process activity are blocked to prevent partitioning and possible data corruption.

Another aspect of node removal is the need to ensure that all I/O requests initiated by the removed node complete prior to the initiation of new I/O requests to the same disks. To enhance disk performance, many disk controllers can reduce head movements by altering the order of simultaneously outstanding commands. This command reordering is not a problem during normal operation; applications initiating I/O requests coordinate with each other using the lock manager, for instance, so that multiple writes, or multiple reads and writes, to the same disk location are never outstanding at the same time. However, when a node fails, all locks held by processes running on that node are released. Releasing these locks allows the granting of locks that are waiting and the initiation of new I/O needs a response based on one of these messages, the application is blocked. Otherwise, the failure does not affect the application. Once the reconfiguration starts, locking is blocked. An application using the

requests. If new locks are granted, a disk controller may move the new I/O requests (issued under the new locks) in front of old I/O requests. To prevent this reordering, a special MSCP command is issued by the connection manager to each disk before new locks are granted. This command creates a barrier for each disk that ensures that all old commands complete prior to the initiation of new commands.

Physical Location and Availability of Disks

The application designer does not generally have to be concerned with the physical location of

a disk in a VAXcluster system. Disks located on HSC storage controllers are directly available to VAX processors on the same CI bus. These disks can then be MSCP-served to any VAX processor that is not connected to that bus. Similarly, disks accessed by way of a local disk controller on a VAX processor can be MSCP-served to all other nodes. This flexibility allows an application to access a disk regardless of physical location. The only differences that the application can detect are varying transfer rates and latencies, which depend on the exact path to the disk and the type of controllers involved.

To provide the best application availability, the following guidelines should be considered:

2. Multiple paths should exist to any given disk. A disk should be dual-pathed between multiple controllers. Dual pathing allows the disk to survive controller failures.

3. Members of the same shadow set should be connected to different controllers or buses as determined by the type of disk.

4. Multiple servers should be used whenever serving disks to a cluster in order to provide continued disk access in the event of a server failure.

Selection of Controllers

Using static load balancing, the VMS software attempts to select the optimal MSCP server for a disk unit when that unit is initially brought on line. The load information provided by the MSCP server is considered in this decision. The HSC controllers do not participate in this algorithm. In addition, the VMS software selects a local controller in preference to a remote MSCP server, where possible. If a remote server is in use and the disk becomes available by way of a local controller, the

1. VMS Volume Shadowing Phase II should be used to shadow disks, thus allowing operations to continue transparently in the event that a single disk fails.

software begins to access the disk though the local controller. This feature is know as local fail-back.

An advanced development effort in the VMS operating system is demonstrating the viability of dynamic load balancing across MSCP servers. Load balancing considers server loading dynamically and moves disk paths between servers to balance the load among the servers.

Disk and Tape Path Changes

Path failures are initially detected by the low-level communication software, i.e., the SCS or port layers. The communications software then notifies the disk or tape class driver of the failure. The driver then transparently blocks the initiation of new I/O requests to the device, prepares to restart outstanding I/O operations, and begins a search for a new path to the device. Static load balancing information is considered when attempting to find a new path. The path search is accomplished by sending an MSCP GET UNIT STATUS command to any known disk controller or MSCP server capable of serving the device. Some consideration is given to selecting the optimal controller; for example, the driver interrogates local controllers before remote controllers.

prevents data corruption in the event that someone substitutes the storage medium without dismounting and remounting the device. After a successful check, the software restarts incomplete I/O requests and allows stalled I/O requests to proceed. In the case of tapes, the tape must be repositioned to the correct location before restarting I/O requests.

If the label check determines that the original medium is no longer on the disk or tape unit, then I/O requests continue to be stalled and a message is sent to the operator requesting manual intervention to correct the problem. Attempts to reestablish the correct operation of a disk or tape continue for an interval determined by the system parameter MVTIMOUT (mount verification time-out). Once the time-out period expires, further attempts to restore are abandoned and pending requests are returned to the application with an error status. Thus, the software handles temporary disk path failures in such a transparent fashion that the application program, e.g., the user application, VAX RMS software, or the VMS file system, is unaware

Once a new path is discovered or the old path reestablished, the VMS system checks the volume label to ensure that the disk or tape volume has not been changed on the device. This verification that an interruption occurred.

Disk Failures

If a disk fails completely when VMS Volume Shadowing Phase II software is used, the software removes the failed disk from the shadow set and satisfies all further I/O requests using a surviving disk. If a block of data cannot be recovered from a disk in a shadow set, the software recovers the data from the corresponding block on another disk, returns the data to the user, and places the data on the bad disk so that subsequent reads will obtain the good data.[2]

Summary

VAXcluster systems continue to provide a unique base for building highly available distributed systems that span a wide range of configurations and usages. In addition, VAXcluster computer systems can grow with an organization. The availability, flexibility, and growth potential of VAXcluster systems result from the ability to add or remove storage and processing components without affecting normal operations.

References

Phase II-Host-based Shadowing," Digital Technical Journal, vol. 3, no. 3 (Summer 1991, this issue): 7-15.

3. W. Snaman, Jr. and D. Thiel, "The VAX/VMS Distributed Lock Manager," Digital Technical Journal, no. 5 (September 1987): 29-44.

4. W. Laing, J. Johnson, and R. Landau, "Transaction Management Support in the VMS Operating System Kernel," Digital Technical Journal, vol. 3. no. 1 (Winter 1991): 33-44.

5. Guidelines for VAXcluster System Configurations (Maynard: Digital Equipment Corporation, Order No. EK-VAXCS-CG-004, 1990).

6. L. Leahy, "New Availability Features of Local Area VAXcluster Systems," Digital Technical Journal, vol. 3, no. 3 (Summer 1991, this issue): 27-35.

7. T.K. Rengarajan, P. Spiro, W. Wright, "High Availability Mechanisms of VAX DBMS Software," Digital Technical

1. N. Kronenberg, H. Levy, and W. Strecker, "VAXclusters: A Closely-coupled Distributed System," ACM Transactions on Computer Systems, vol. 4, no. 2 (May 1986): 130-146.

2. S. Davis, "Design of VMS Volume Shadowing Journal, no. 8 (February 1989): 88-98.

8. A. Goldstein, "The Design and Implementation of a Distributed File System," Digital Technical Journal, no. 5 (September 1987): 45-55.

===
Copyright 1991 Digital Equipment Corporation. Forwarding and copying of this article is permitted for personal and educational purposes without fee provided that Digital Equipment Corporation's copyright is retained with the article and that the content is not modified. This article is not to be distributed for commercial advantage. Abstracting with credit of Digital Equipment Corporation's authorship is permitted. All rights reserved.
===