VAXcluster Software Technical Summary

digital

VAXcluster Software Technical Summary

Table of Contents

Introduction

VAXcluster systems are the heart of a new concept in VAX/VMS computing. Clustering of VAX computers places greater computing power, a larger computer system, and more resources and data at your fingertips. Whether you are a VAX owner looking for ways to extend your computer system or are just considering how VAX computing could help your organization, the VAXcluster approach could be the perfect answer to your computing needs. It enables you to provide computer resources and applications to your entire organization as well as to the individual user. What's more, you can deal with expansion and change in a productive and controllable manner. Without needing to acquire new skills or change your present work patterns, you can gain greater access to information stored in databases throughout your company and improve your ability to share this information over a larger network.

The *VMS V4.0 VAXcluster Software Technical Summary* explains what a VAXcluster system is, how it operates, and how your applications can benefit from it.

The first two chapters describe a VAXcluster system and the benefits it provides.

Chapter 3 discusses the hardware components that make up a cluster; an entire section is devoted to the Terminal Server.

Chapter 4 highlights the features of the hardware and Version 4.0 of the VMS operating system.

Chapter 5 is a description of the major software pieces that were engineered in VMS V4.0 for VAXclusters.

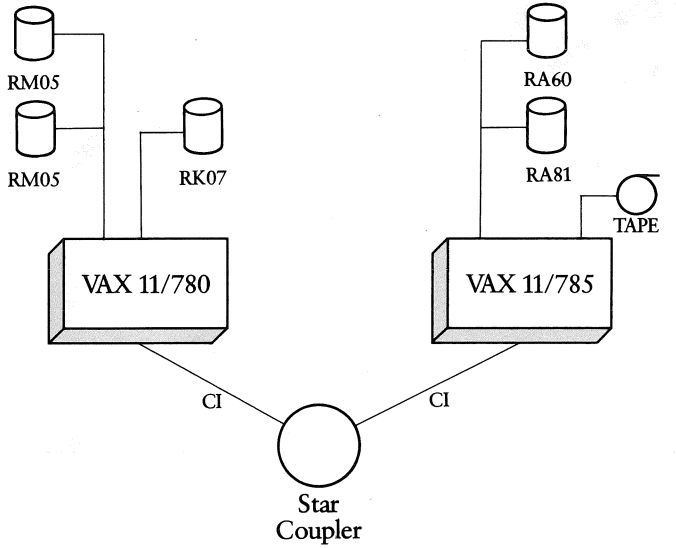
Chapter 6 discusses system management of a VAXcluster system from the system manager's viewpoint.

Chapter 7 gives an overview of VAXcluster capabilities that are yet to come.

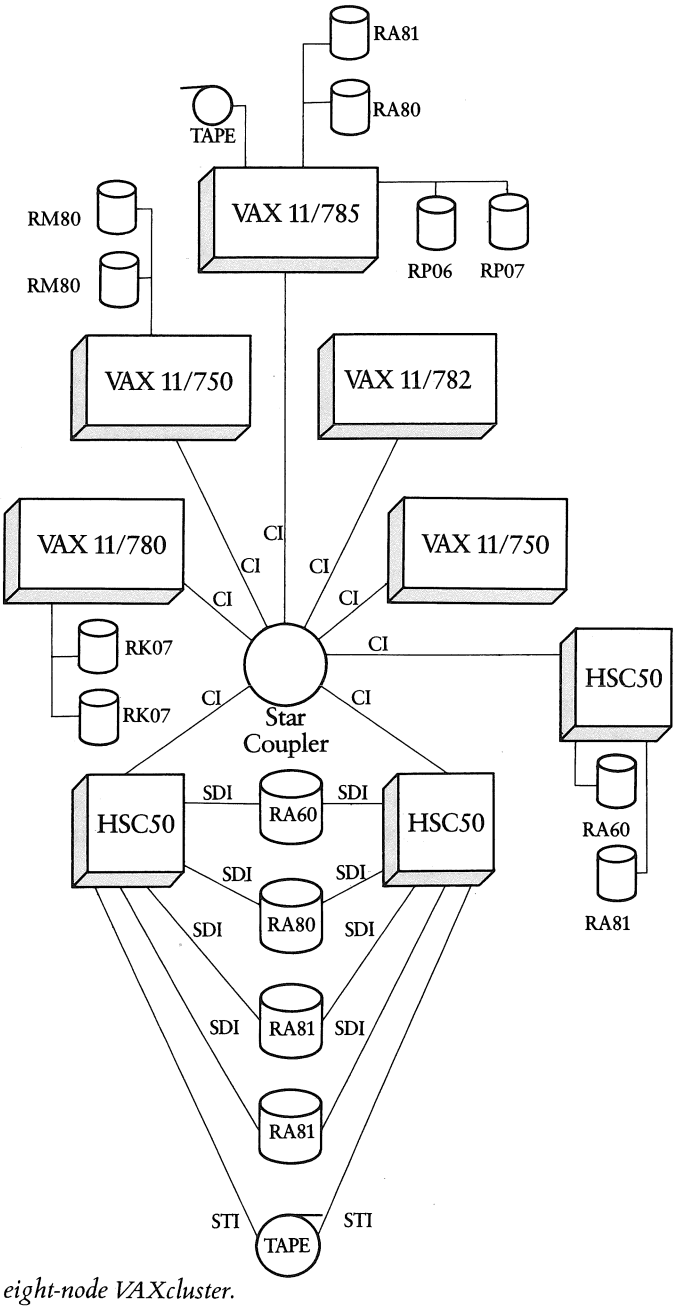
Appendix I updates a previously published table showing the VAXcluster capabilities initially provided by phase one of the VAXcluster program through VMS Version 3, those added by VMS V4.0 in phase two of the program, and items that will be implemented in future VMS releases.

Appendix II diagrammatically illustrates the VAXcluster hardware and software components.

A VAXcluster configuration is a system that combines two or more VAX processors, and mass storage servers if desired, in a loosely coupled manner. Clustering is possible with VAX-11/750, VAX-11/780, VAX-11/782, VAX-11/785, and VAX 8600 processors. The mass storage server is a free-standing, high-speed, intelligent device designed to the specifications of the Digital Storage Architecture and known as the Hierarchical Storage Controller, or HSC50. Each VAX processor or HSC50 in a cluster is called a node. Up to 16 nodes may be connected in a VAXcluster via a high-speed bus known as the Computer Interconnect, or CI.



A two-node VAXcluster.



An eight-node VAXcluster.

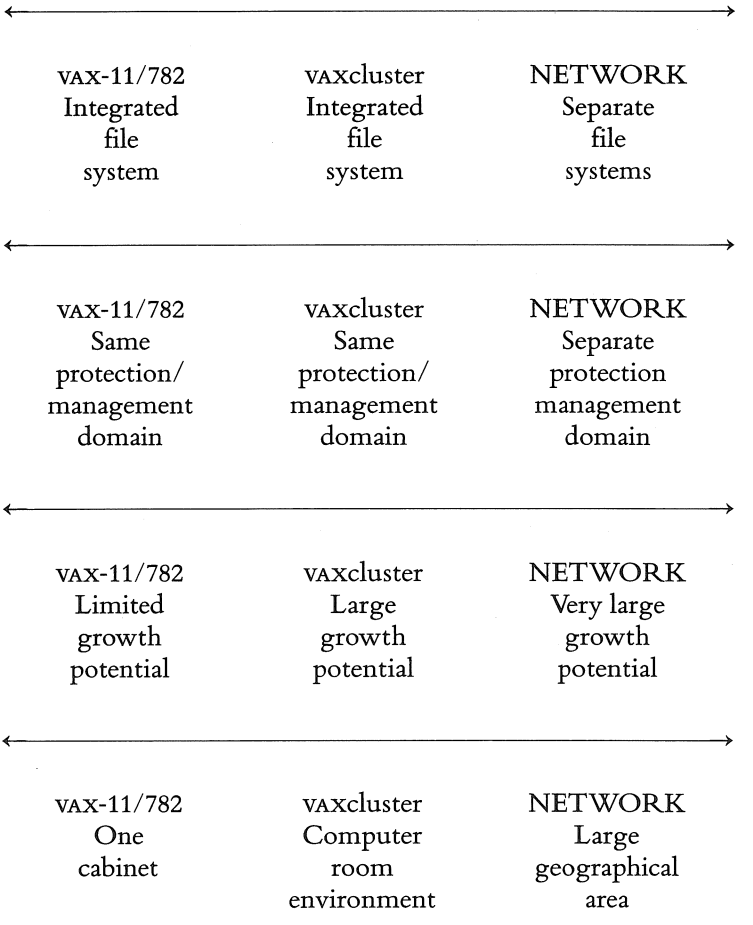
One way to look at multiprocessing systems is to consider a spectrum of systems with two extremes. At one end of the spectrum would be a very tightly coupled multiprocessing system. Within the VAX family, it would be the VAX-11/782 system—a tightly coupled multiprocessor characterized by one copy of the VMS operating system residing in shared memory between two processors. At the other end of the spectrum is the computer network. The network is a very loosely coupled multiprocessing system in which each processor has its own copy of the operating system. The network systems can be tied together by DECnet software using point-to-point, X.25, or Ethernet connections.

VAX-11/782 NETWORK

The VAXcluster architecture combines the advantages of both types of multiprocessing.

The diagrams below show characteristics that were chosen to endow VAXcluster systems with the advantages found in the spectrum of multiprocessing systems.

VAX-11/782	VAXcluster	NETWORK
Systems boot and fail together	Systems boot and fail independently	Systems boot and fail independently


VAX-11/782
Integrated
file
system
VAXcluster
Integrated
file
system
NETWORK
Separate
file
systems
VAX-11/782
Same
protection/
management
domain
VAXcluster
Same
protection/
management
domain
NETWORK
Separate
protection
management
domain
VAX-11/782
Limited
growth
potential
VAXcluster
Large
growth
potential
NETWORK
Very large
growth
potential
VAX-11/782
One
cabinet
VAXcluster
Computer
room
environment
NETWORK
Large
geographical
area

Highlights of a VAXcluster

VAXcluster configurations may grow system-by-system to meet the increased demand of users. As an organization's needs grow, more computing or I/O resources can be plugged into the VAXcluster without interrupting its operation.

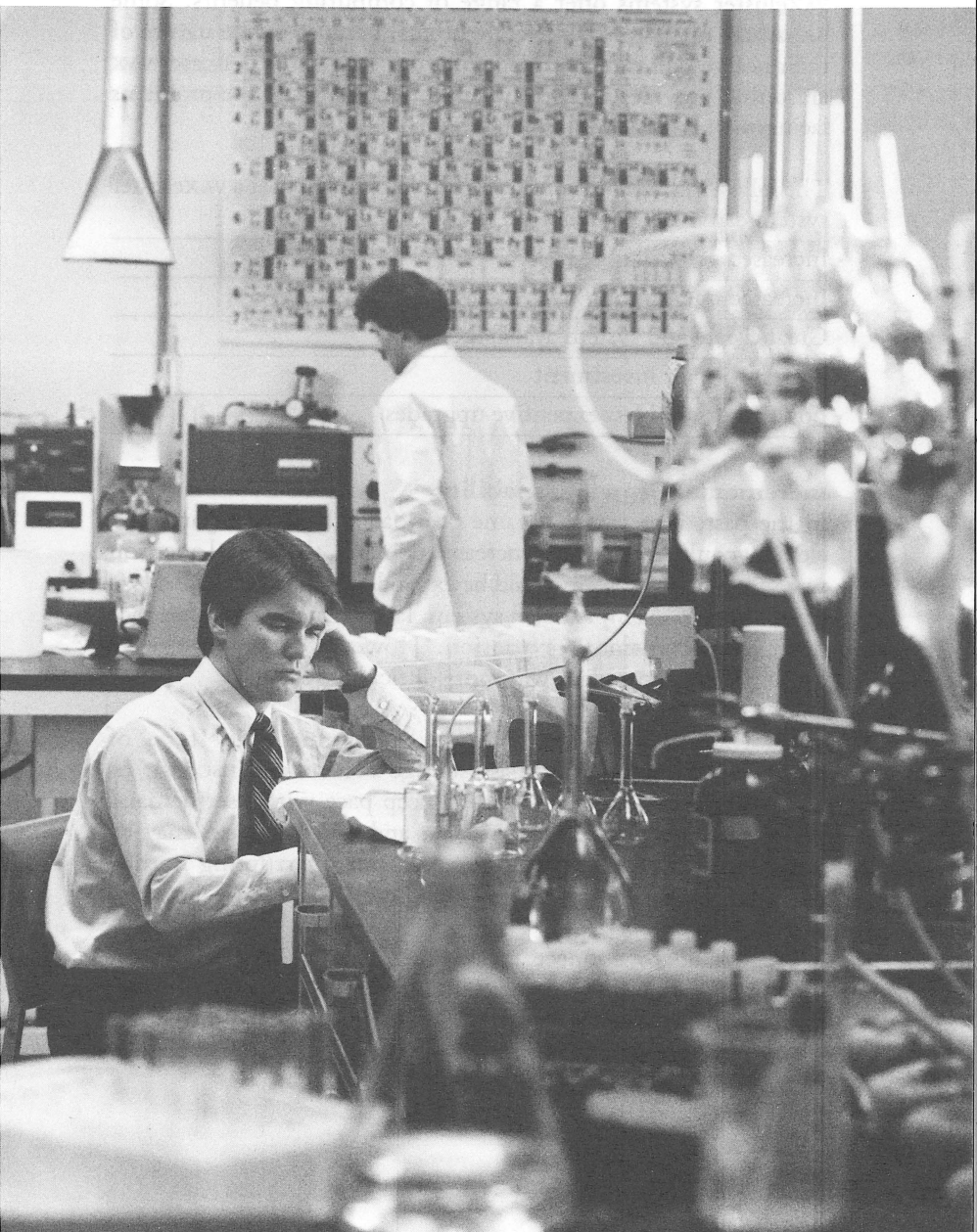
All VAXcluster disks, whether connected to a VAX processor directly or through an HSC50 controller, may be made available to a user logged on to any VAX node. Data on any VAXcluster disk may be shared at the volume, file, and record levels by users on either the same VAX processor or different ones.

Batch and print jobs may be load-balanced throughout the VAXcluster.

Redundancy features built into the hardware combined with failover capabilities of the software provide greater system and data availability.

These features and others are discussed in greater detail in subsequent sections.

Chapter 2 · VAXcluster BENEFITS



VAXcluster systems offer a range of computing benefits. Some users may use all of them while others, because of the nature of their needs, will take advantage of only some. In either case, VAX clustering provides an effective answer for many of the problems facing managers of computer systems.

The following are some of the high-level benefits that a VAXcluster system can bring to users:

- Increased growth potential
- Increased data sharing
- Greater data and system availability
- Preservation of investment
- Lower costs and less expensive upgrades

Incremental Growth Capability

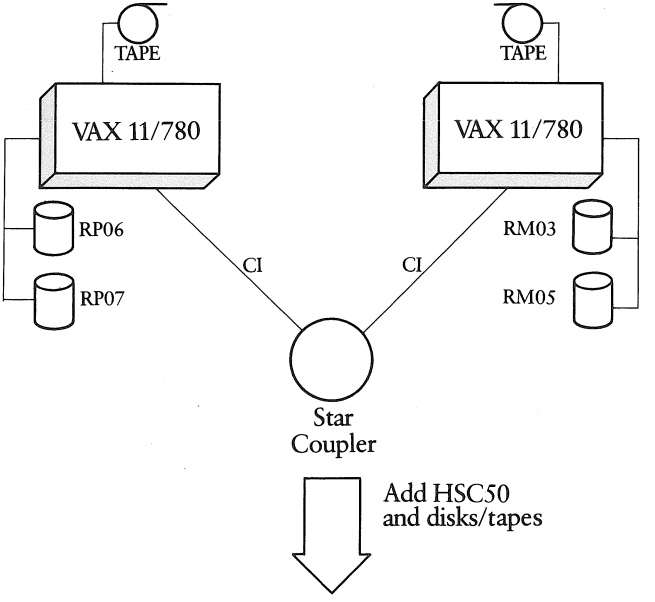
In the past, if a system became overloaded because a particular element could not meet the increased demand, the crucial question was whether the system could be expanded to handle the load. It could if, for example, the system lacked memory but space remained for memory expansion. However, if the CPU were overloaded, the problem was more difficult to solve.

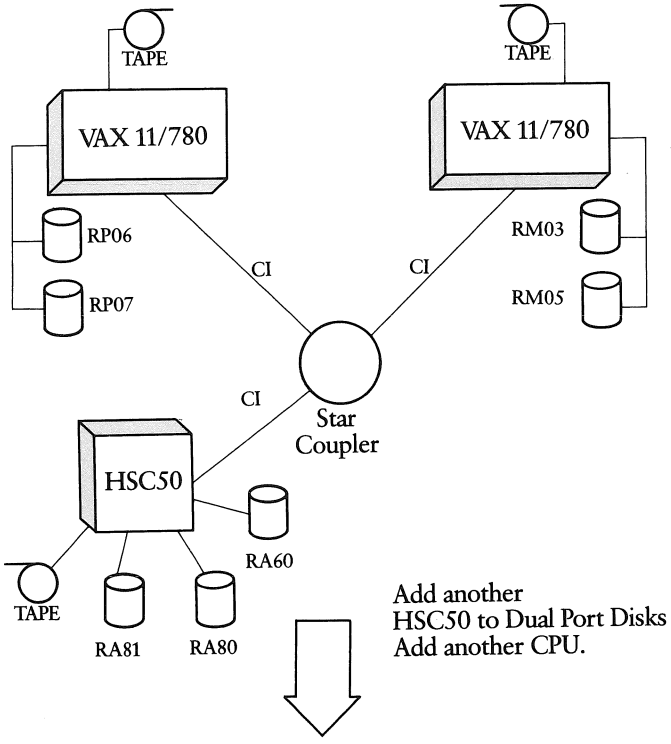
The VAXcluster system solves the dilemma. Any type of resource—memory, CPU, mass storage controllers (HSC50s), or disks and tapes—may be added to keep pace with user demand. More processors, HSC50 controllers, and disks or tapes may be added to an operating VAXcluster without interrupting normal operations.

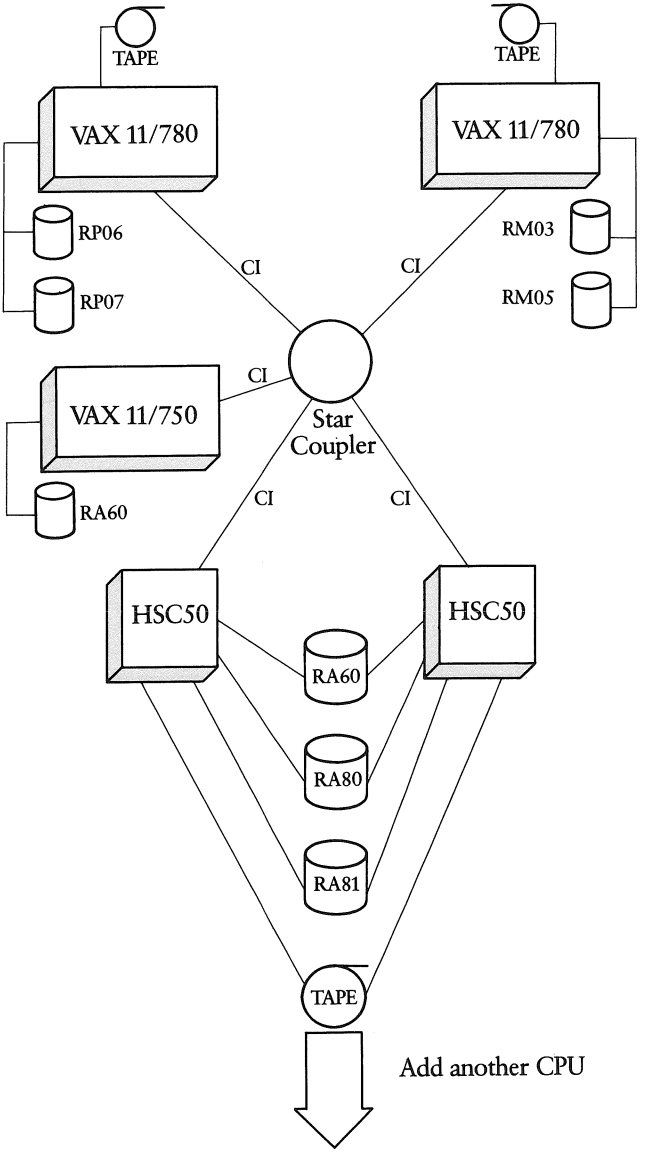
It is possible to connect up to 16 nodes by clustering VAX processors, alone or in combination with HSC50 devices. Moreover, each HSC50 can support up to 24 disks or tape formatters. That means you can create very large systems, either all at once or system by system.

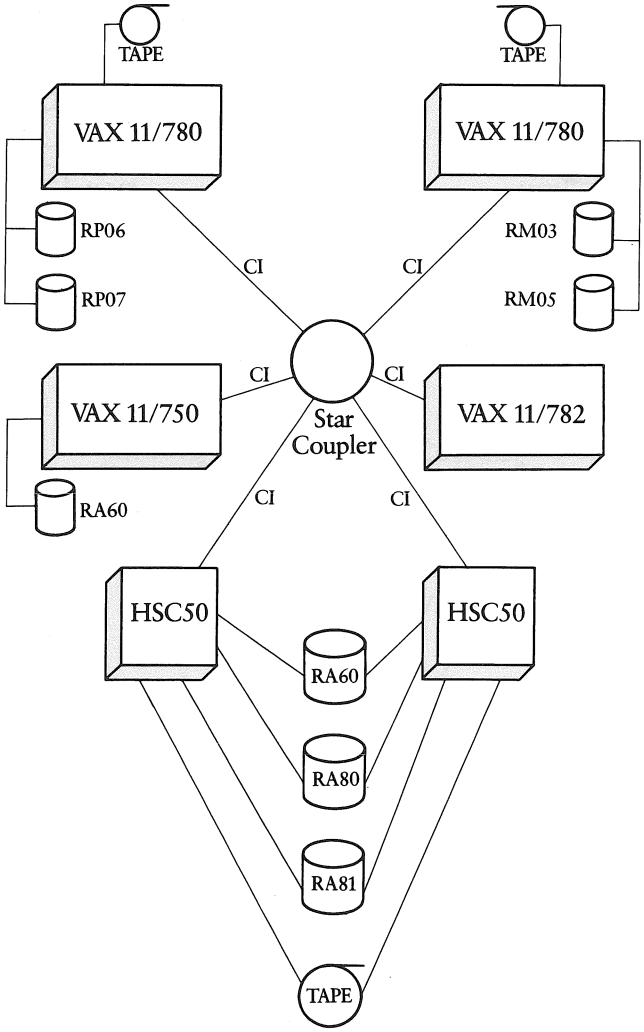
Because the demands on most computer systems increase with time, the VAXcluster system is an elegant response to that growing demand. You can start at an early stage because as few as two connected VAX systems constitute a VAXcluster. But it ultimately can grow into a very large system containing many VAX systems and mass storage controllers.

The growth of a VAXcluster over time, starting with a simple two-node cluster and growing to a much bigger cluster with HSC50's.



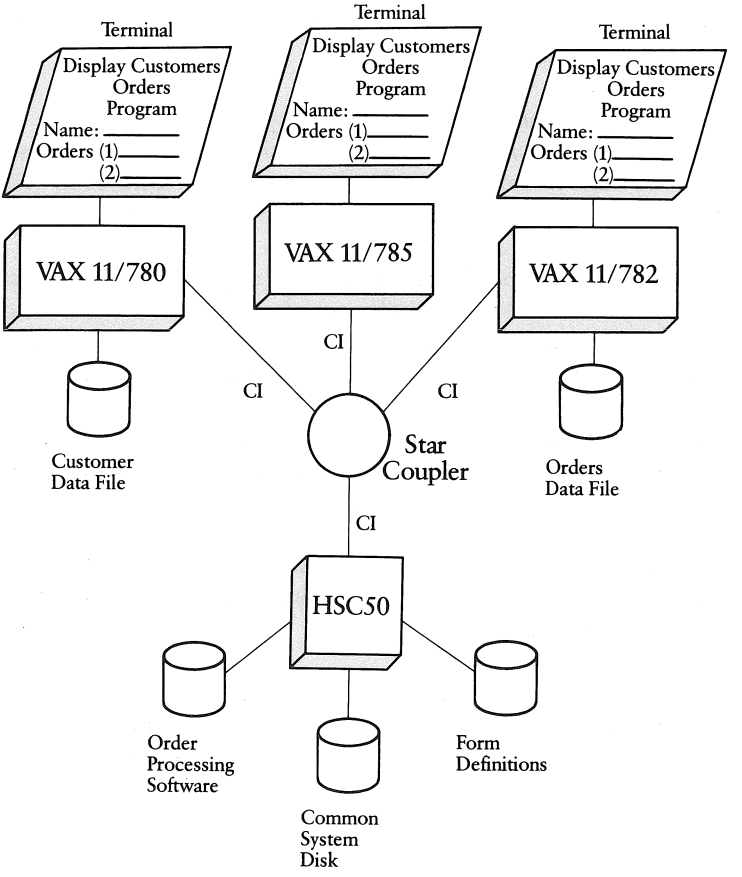

TAPE
VAX 11/780
RP06
RP07
CI
TAPE
VAX 11/780
RM03
RM05
CI
CI
Star
Coupler
HSC50
RA60
TAPE
RA81
RA80
Add another
HSC50 to Dual Port Disks
Add another CPU.


TAPE
TAPE
VAX 11/780
VAX 11/780
RP06
RP07
RM03
RM05
CI
CI
CI
VAX 11/750
Star
Coupler
RA60
CI
CI
HSC50
HSC50
RA60
RA80
RA81
TAPE
Add another CPU


TAPE
TAPE
VAX 11/780
VAX 11/780
RP06
RP07
RM03
RM05
CI
CI
CI
CI
VAX 11/750
VAX 11/782
Star
Coupler
RA60
CI
CI
HSC50
HSC50
RA60
RA80
RA81
TAPE

Increased Data-Sharing

At the system manager's discretion, a VAXcluster user can have access to disks connected locally to any VAX processor or any HSC50 in the cluster. Files on those disks may be shared at the record level by all VAXcluster users, retaining full read-and-write capability.



A single application spread across an entire VAXcluster system.

This new capability is a major step forward in operating-system software technology and allows for increased data-sharing. Applications may be spread across multiple processors, or users of one application may operate across multiple processors. The system manager may choose to set up a homogeneous VAXcluster environment, in which users need not be concerned about which VAX processor they are using because each processor presents an equivalent environment.

Greater Data and System Availability

Data in a VAXcluster system is more accessible and more secure as a result of built-in hardware redundancy, the new features in VMS V4.0, and the presence of several VAX processors in the VAXcluster.

Hardware redundancy is inherent in the computer-interconnect (CI) data path; in the Star Coupler, which is the central connecting point of all nodes; and in the ability to connect two HSC50s to each Digital Storage Architecture (DSA) disk or tape in the cluster.

One VMS V4.0 software feature that enhances data availability is automatic failover for a dual-ported disk drive if its HSC50 fails. The failover is transparent to the VAXcluster users.

If a processor in a VAXcluster fails, VMS V4.0 releases all the locks that the failed processor has on resources so the remaining VAX nodes can continue to work.

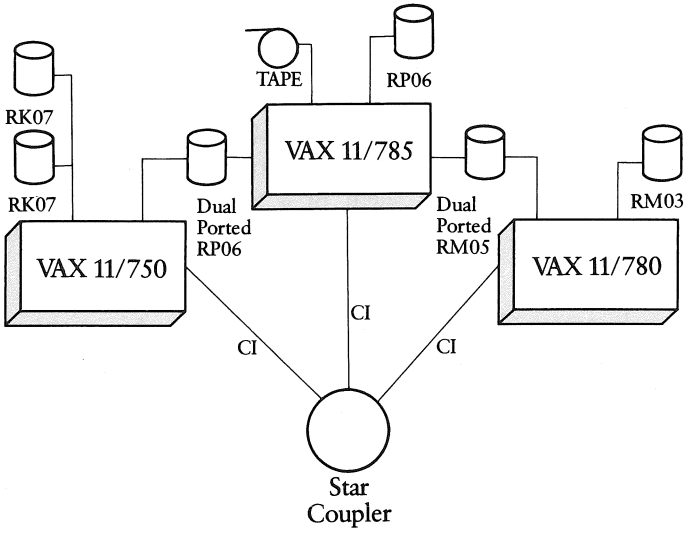
These hardware and software features permit configuration of systems that provide greater system and data availability because processing can continue despite certain system failures.

Preserved Investment

The VAXcluster concept, as a state-of-the-art method of configuring computers, is designed to support the newest hardware and software, such as the CI bus, HSC50 mass storage controllers, and Digital Storage Architecture (DSA) disks and tapes. However, customers who have invested in UNIBUS and MASSBUS disks may desire to continue using them in a VAXcluster environment, thereby preserving their investment. They can, thanks to software in VMS V4.0.

A VAXcluster may be configured without HSC50s. Any UNIBUS or MASSBUS disk (which, by definition, is connected locally to a VAX processor) may be shared by all users in a VAXcluster. The system manager determines which local disks will be accessible to all cluster nodes. VAXcluster systems also permit full read-and-write access to dual-ported MASSBUS disks along both access paths simultaneously. Other nodes in the cluster may access a dual-ported MASSBUS disk using any available path, with fully automatic failover, should either of the directly connected processors fail.

In this way, the state-of-the-art software allows VAXcluster configurations that contain UNIBUS or MASSBUS disks, preserving and enhancing customers' investment in earlier disk technology.



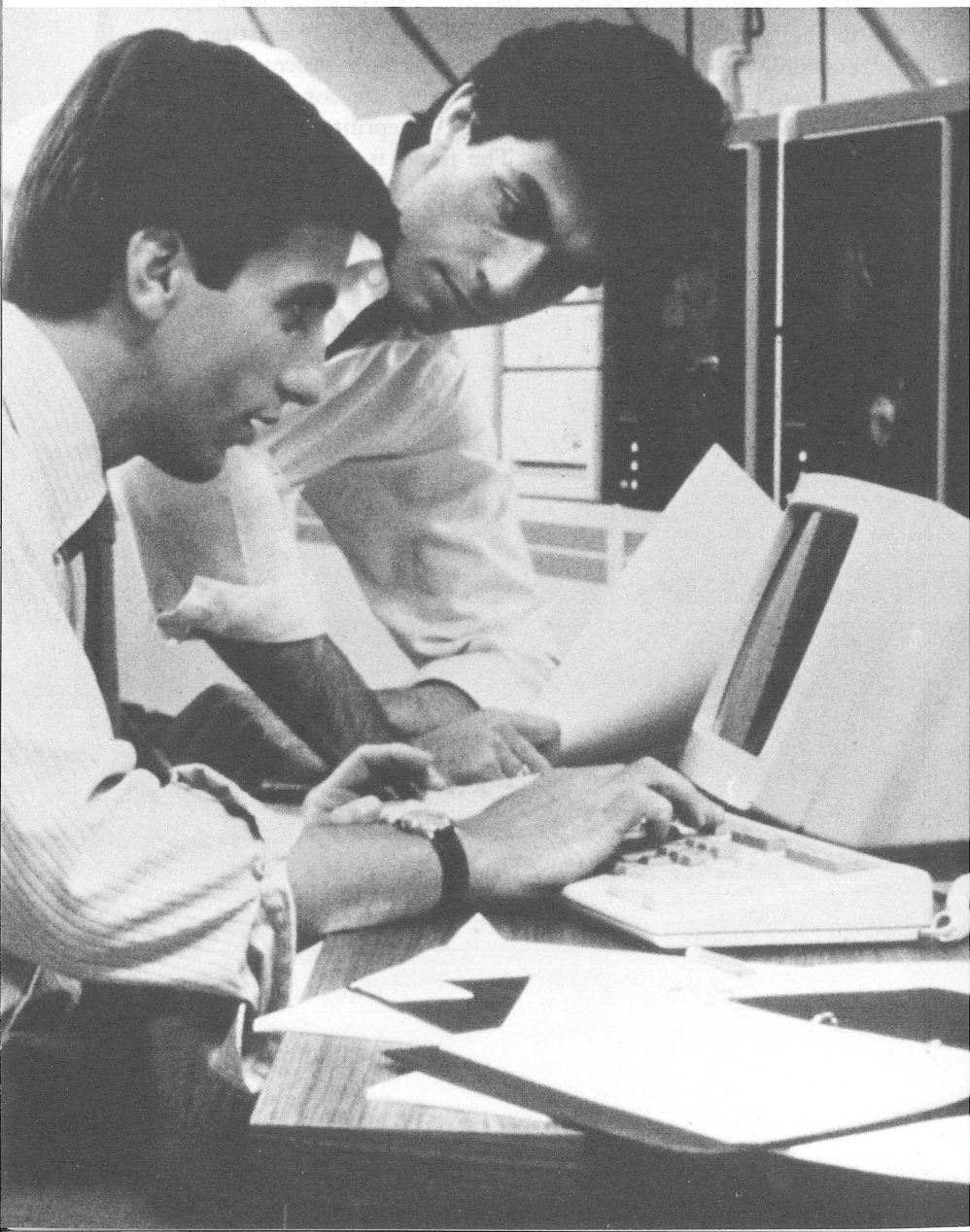
A VAXcluster with MASSBUS and UNIBUS disks, some of which are dual-ported.

Lower Expansion Costs

The VAXcluster approach can reduce the cost of expanding your system when a hardware component no longer meets user demands. In the past, if the limiting component was the VAX processor itself, for example, then another CPU had to be purchased—along with its own system disk, I/O controllers, and possibly more terminals. With the VAXcluster, the only necessary purchases are another processor, CI interface, and memory. No longer is there any need to buy a separate system disk, data disks, terminals, or printers for the new processor.

The new system may be booted from an existing disk on an HSC50 in the VAXcluster, and programs and data already existing may continue to be used in the cluster. The new processor can be accessed from existing terminals if they are connected to the VAXcluster through a terminal server. Lineprinters already in the VAXcluster also may be used. The result is lower cost because additional hardware previously required to expand a system often is not needed when a VAXcluster is enlarged.

Chapter 3· VAXcluster HARDWARE COMPONENTS



A VAXcluster can start either with the purchase of a VAX computer or with your present VAX-11/750, VAX-11/780, VAX-11/782, or VAX-11/785 system.

A VAXcluster configuration contains four elements in addition to the one or more VAX/VMS processors, mass storage devices, and terminal communications capabilities. The four additional elements are the CI (Computer Interconnect) dual-path, high-bandwidth bus for interconnecting cluster nodes; the CI750 and CI780 intelligent CI port controllers; the HSC50 (Hierarchical Storage Controller) mass storage server; and the Star Coupler, which serves as the central connection point for all cluster nodes.

This section concerns these four hardware elements of a VAXcluster and their interrelationships.

The Computer Interconnect

The CI (Computer Interconnect) is a high-speed, fault-tolerant, dual-path bus. Using a CI bus, any combination of processor nodes and intelligent I/O-subsystem nodes up to 16 in number can be loosely coupled in a computer-room environment.

Nodes in a VAXcluster use a multiaccess-bus topology that allows any VAX node in the cluster to talk to any other VAX node. Either a VAX processor or an intelligent I/O subsystem such as a HSC50 can serve as a node. The VMS operating system uses a new system-level protocol to communicate among cluster nodes.

The CI bus, which has a bandwidth of 70 Mbits per second, features an immediate acknowledgment scheme by which channel time is reserved at the end of each message to allow the destination device to acknowledge receipt of the message.

The loss of any node from the cluster does not block communication among the remaining nodes because no single node is bus master.

The CI780 and CI750 Interfaces

The CI interfaces are microcoded intelligent controllers that connect VAX-11/780, VAX-11/782, VAX-11/785, and VAX-11/750 processors to the CI bus. Each interface attaches to one CI bus, which consists of two transmit cables and two receive cables. Under normal operating conditions, both sets of cables are available to meet traffic demands. If one path becomes unavailable, then all traffic uses the remaining path. Meanwhile, the VMS operating system periodically tests a failed path. As soon as the path becomes available again, it is automatically used for normal traffic.

The Star Coupler

The Star Coupler is the common connection point for all cluster nodes linked to the CI. It connects all CI cables from the individual nodes, creating a radial or star arrangement that has a maximum radius of 45 meters. The Star Coupler can be configured to support VAXcluster systems of up to 16 nodes.

The Star Coupler provides passive coupling of the signals from all cluster nodes by means of power splitter/combiner transformers, permitting the removal of any node from the cluster without affecting the operation of any other node. In addition, the dual paths of the CI bus are electrically separated from each other.

For each node, transmit and receive connectors are provided for each CI path. A signal received from a transmit cable is distributed to all receive cables. The Star Coupler terminates all cables with their characteristic impedance. This allows connection or disconnection of nodes during normal cluster operations without affecting the rest of the cluster.

Figure 3.1 shows the electrical topology of cluster nodes with respect to the Star Coupler. Figure 3.2 shows the logical topology of the cluster.

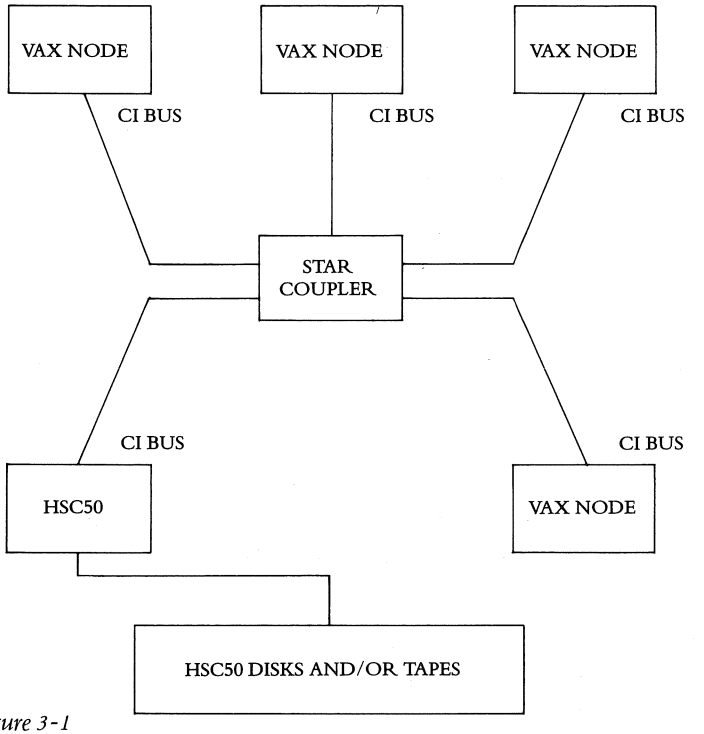


Figure 3-1
Cluster Node-to-Star Coupler
Electrical Topology.

The HSC50

The HSC50 (Hierarchical Storage Controller) is a self-contained, intelligent, mass storage subsystem that connects one or more host processors to a set of mass storage disks or tapes. The HSC50, itself a cluster node, communicates with host CPUs by way of the

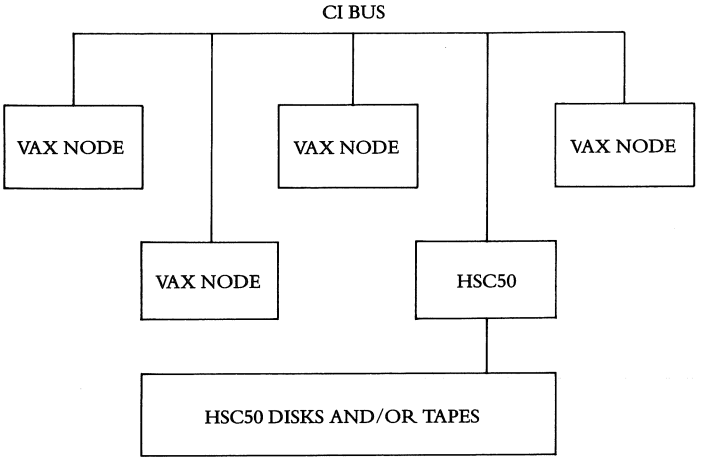


Figure 3-2
VAXcluster Multiaccess Logical Topology.

CI, using Digital's MSCP (Mass Storage Control Protocol) for host communications. Communication between the HSC50 and the mass storage drives is through the Standard Disk Interface (SDI) and the Standard Tape Interface (STI).

To maximize throughput, the HSC50 handles two or more concurrent operations on several drives in order to optimize physical operations such as track seeks and rotational positioning.

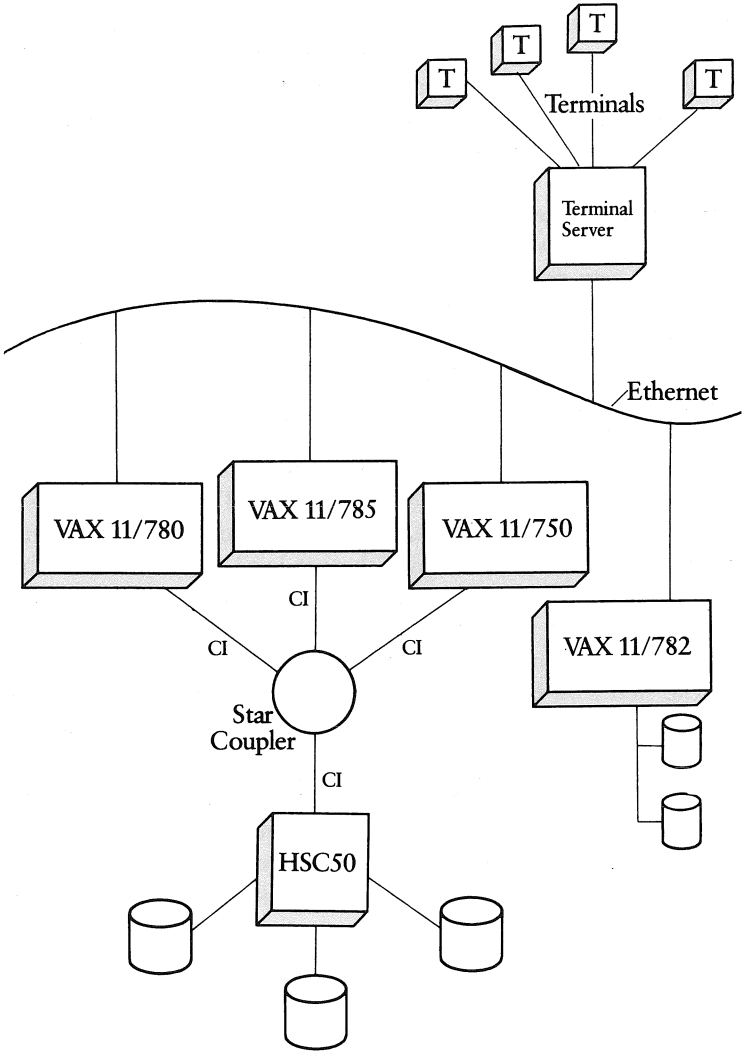
The HSC50 subsystem has a comprehensive set of autonomous diagnostics and redundant components. It also can continue operating in a degraded mode with minimal reduction in throughput if a disk or tape interface module fails. In addition, the SDI and STI also use passive coupling, so they can be disconnected and reconnected without disrupting the operation of other VAXcluster devices.

Each HSC50 can support up to six interfaces made up of any combination of SDIs and STIs. An SDI can support four HSC50-compatible disks, yielding a maximum of 24 disks per HSC50. And a STI can support four masters plus three slaves per master, resulting in a maximum of 96 tape drives per HSC50.

Terminal Servers

VMS V4.0 supports Digital's Ethernet Terminal Server products, both the new packaged system, which includes hardware and software, and servers that users configure with the LAT-11 software product, for their existing PDP-11 systems. Note that all software features supported by the LAT-11 package are also supported by the software for the Ethernet Terminal Server packaged system. With either Terminal Server, a terminal user can establish a session between the terminal and a VMS host node connected to the same Ethernet and running the LAT protocol. The LAT protocol is a specially designed set of rules and procedures for connecting terminals to host nodes in an Ethernet local area network. The terminal I/O will then be essentially indistinguishable from I/O using a directly connected terminal. However, unlike direct terminal connections, the Terminal Server gives users easy access to all VMS nodes on Ethernet that run the LAT protocol. In addition, Terminal Servers that use the LAT protocol give users a variety of unique benefits, as follows:

- Terminal Connection Management. Each terminal connected to the Terminal Server can connect to the same node or a different one. Furthermore, several terminal servers can be connected to the same Ethernet, permitting many terminals to access one or more nodes on Ethernet.



The Terminal Server connected to VMS nodes via Ethernet.

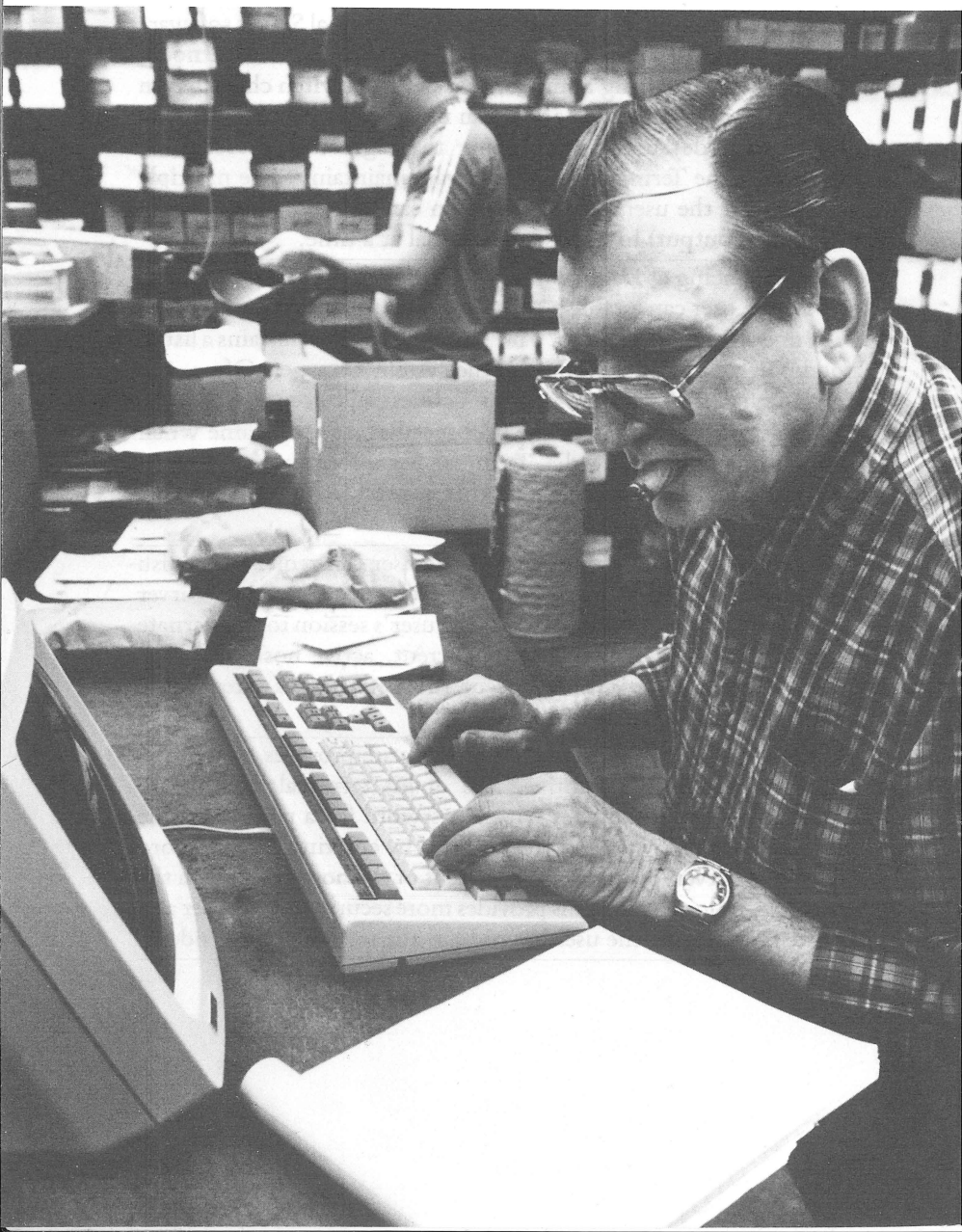
- Multiple Simultaneous Virtual Circuits. Terminal Server software allows a user to establish and maintain sessions to several nodes simultaneously. Using a simple, user-defined switch character on the terminal keyboard, the user can move from session to session (node to node) without repeating the login dialog each time. Although the Terminal Server software maintains these multiple sessions for the user, only one session can be active (performing input and output) to the user's terminal at a time.

- Load Balancing at Login Time. When a user wishes to login to a node, the user can establish a session either to a specific node or to any one of a group of nodes. The Terminal Server maintains a list of nodes in each group defined by the system manager. Often, the group that the system manager defines will comprise all the systems in a VAXcluster. If the user specifies a group name when establishing a session, the Terminal Server software creates a session to the node in the group that appears to have the greatest available computing capacity.

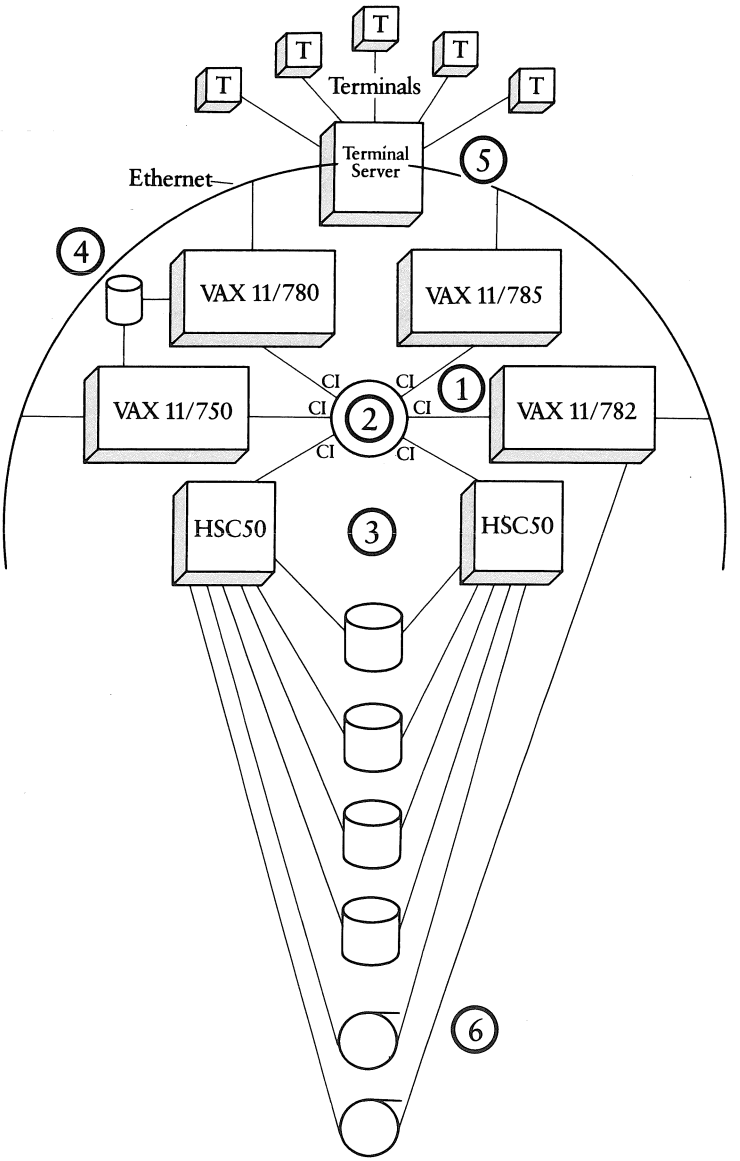
- Automatic Login Failover. Just as the user can request establishment of a session to any node in a group, the Terminal Server software can automatically switch the user's session to an alternate node in that group if the user's current, active host node fails. Should this automatic failover occur, the user only has to repeat the login dialogue to resume computing.

- Terminal Locking. Terminal Server software provides a way for a user to prevent unauthorized use of his terminal; that is, to logically lock the terminal with a password for a period of time. The user later may unlock the terminal by entering the password. Terminal Server locking is independent of the nodes on which the user may be logged in. This provides more security for the user and his data whenever the user has to leave the terminal unattended.

Chapter 4· SYSTEM AVAILABILITY



The following diagram shows a fully configured VAXcluster. Each numeric label is described below to indicate the added availability the particular feature brings to the VAXcluster.



The Computer Interconnect (1) has built-in redundancy. It is a dual-path bus. Under normal operations both paths can be used; however, if one of the paths fails, communication can still occur over the remaining path. The VMS operating system periodically tests a failed path and puts it back online once the failure has been repaired.

The Star Coupler (2) also has built-in redundancy. A CI consists of four coaxial cables, two to transmit and two to receive. Inside the Star Coupler are two transformers for every eight nodes. One transmit cable and one receive cable are connected to the first transformer, and the remaining transmit cable and receive cable are connected to the second transformer. Every HSC50 or VAX processor node connects its CI in this way. In the very unlikely event that a transformer inside the Star Coupler fails, communication can still occur via the remaining transformer.

Digital Storage Architecture (DSA) disks and tapes may be connected between two HSC50 controllers (3). If one of the HSC50s fails, the VMS operating system will failover to the second HSC50, transparently to the user. Under normal operation, disks and tapes may be manually load-balanced between both HSC50s. If one of the HSC50s fails, the disks that were online to the failing HSC50 then become online through the other HSC50. Even though DSA devices can be physically connected to two controllers, only one of the controllers has an active path to the device for data transfer. On a failover, the active path may change to a different controller. But since a device can have only one active path at a time, dynamic dual-porting, where I/O occurs simultaneously from both controllers, is not available with DSA controllers. However, this is available with MASSBUS disks.

Dual-ported MASSBUS disks (4) are supported in VMS V4.0, with full read-and-write capability from both VAX processors simultaneously. However, the dual-ported MASSBUS disk cannot be a VMS system disk, and a CI must be present so that the distributed file system and distributed lock manager may coordinate file and record access on the dual-ported MASSBUS drive.

The terminal server software (5) allows a user to have two or more simultaneous sessions on different nodes. The automatic login failover switches a user's terminal to an alternate node in the cluster if the user's current node fails. The terminal server software also provides load balancing at login time by connecting the user to the least-loaded VAX processor in the cluster.

A DSA disk or tape may be dual-ported between two UDA50s or a UDA50 and an HSC50 (6). However, in either of the configurations, if one of the active paths fails, failover must be performed manually. The system manager must dismount the drive, press the port buttons on the drive, and remount the drive to connect it to the alternate controller.

VAX Processor Failures

In the event that a VAX processor in the cluster fails, all the resources that the failed processor had locked are released. This is a function of the distributed lock manager. When the system detects that a node in the cluster has failed, the remaining VAX nodes temporarily suspend normal operation and the distributed lock manager releases all the locks held by the failed VAX processor. Once this is complete, processing will continue normally on each remaining VAX node as long as the cluster still has enough VOTES to satisfy the QUORUM algorithm requirements (see chapter 4).

Failure of a VAX processor can be detected in two ways. If the reason for the failure of the VAX processor is a software or hardware condition that results in the BUGCHK code being executed, or the system is shutdown by an operator, a datagram is sent out telling the remaining nodes of the failure. If the CPU is halted or, for some reason, the datagram is unsuccessfully transmitted (the CI port has ceased to function), the failed VAX node is detected by software polling, which occurs at an interval selected by the system manager. There are also parameters that control the number of nodes polled and, on detecting a failure, determine how long to wait in case it is merely a transient failure.

Chapter 5· VMS V4.0 AND THE VAXcluster

A range of new VMS components, from class and port drivers to the cluster server process, were engineered for VMS Version 4.0 in order to support the VAXcluster. This chapter describes new components along with some of the lower-level components that have been available since VMS V3.3. It also includes discussions of DECnet in a VAXcluster, the QUORUM algorithm and its impact on system configurations, device naming, and VMS features that are not supported across the VAXcluster.

The following VMS V4.0 components will be highlighted:

- Digital Storage Architecture (DSA) Class and Port Drivers
- System Communication Services
- MSCP (Mass Storage Control Protocol) Server
- Connection Manager
- Distributed Lock Manager
- Distributed File System and RMS
- Distributed Job Controller
- Cluster Server Process

DSA Class and Port Drivers

The Digital Storage Architecture (DSA), which is Digital's latest disk-and-tape technology, is called an architecture because the DSA disk/tape controller has clearly defined functions. A protocol known as the Mass Storage Control Protocol, or MSCP, has been defined between the host processor and the controller. Because of its architectural framework, it is very easy to add new devices and controllers to a VAXcluster. Two of Digital's controllers that adhere to the architecture are the UDA50 and the HSC50, both of which understand the MSCP protocol.

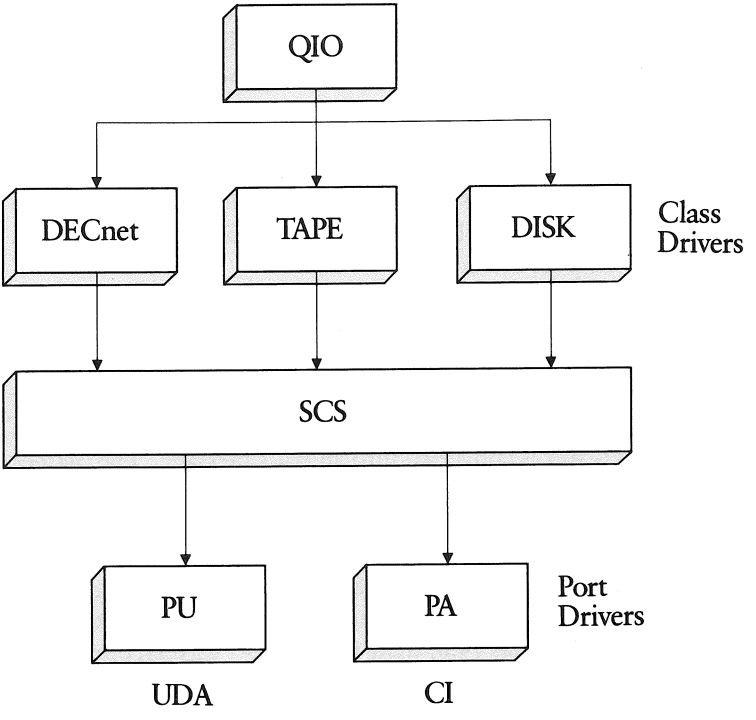
To support the new architecture, VMS includes device drivers for the DSA devices. Unlike a traditional device driver, the driver for the DSA device has two parts, the class and port drivers.

The class driver can be considered to be the generic driver handling all QIO (queue input/output) requests. The disk class driver accepts all disk QIOs for DSA disks; however, it doesn't "know" the physical port through which the I/O will ultimately pass. This is the job of the port driver.

With VMS V4.0, there are three class drivers: the disk class driver (already mentioned in connection with DSA disks), a tape class driver for DSA tapes, and a DECnet class driver for DECnet I/O over the CI. In the case of the tape and disk class drivers, the QIO received by the class driver is converted into an MSCP message. The DECnet class driver converts its QIOs into messages that adhere to the DECnet protocol.

The port driver understands the physical port and nothing else. It ensures that data passes through the port successfully or returns an error status. With VMS V4.0, there are two port drivers, one for the UDA50 and one for the CI780 or CI750, the CI controllers.

An I/O request from a user process to a disk connected to the UDA50 passes through the disk class driver and the UDA50 port driver. An I/O request to a disk connected to an HSC50 passes through the disk class driver and then the CI780/CI750 port driver. Therefore, any future hardware controllers that Digital engineers for DSA devices will require, at most, only a new port driver, not a new class driver. In addition, the same port driver, the CI port driver, may be used by a number of class drivers to handle different classes of I/O requests. For example, the CI port driver would be used by the disk, tape, and DECnet class drivers.



The relationship between port and class drivers.

System Communication Services

System Communication Services, or SCS, is a layer of software between the class driver and port driver. The CI supports three types of data transfers: datagram, sequenced message, and block data transfer. The datagram is used by DECnet, the sequenced message is used by the VMS system for short messages between nodes—for example, distributed-lock-manager requests—and the block data transfer is used for data movement between the cluster nodes, such as input and output by disks and tapes. Consequently, it can be seen that the CI hardware was designed with consideration of the kind of data transfers required by VAXclusters. Therefore, the different modes of data transfer available to the VMS operating system provide efficient data transfers over the CI.

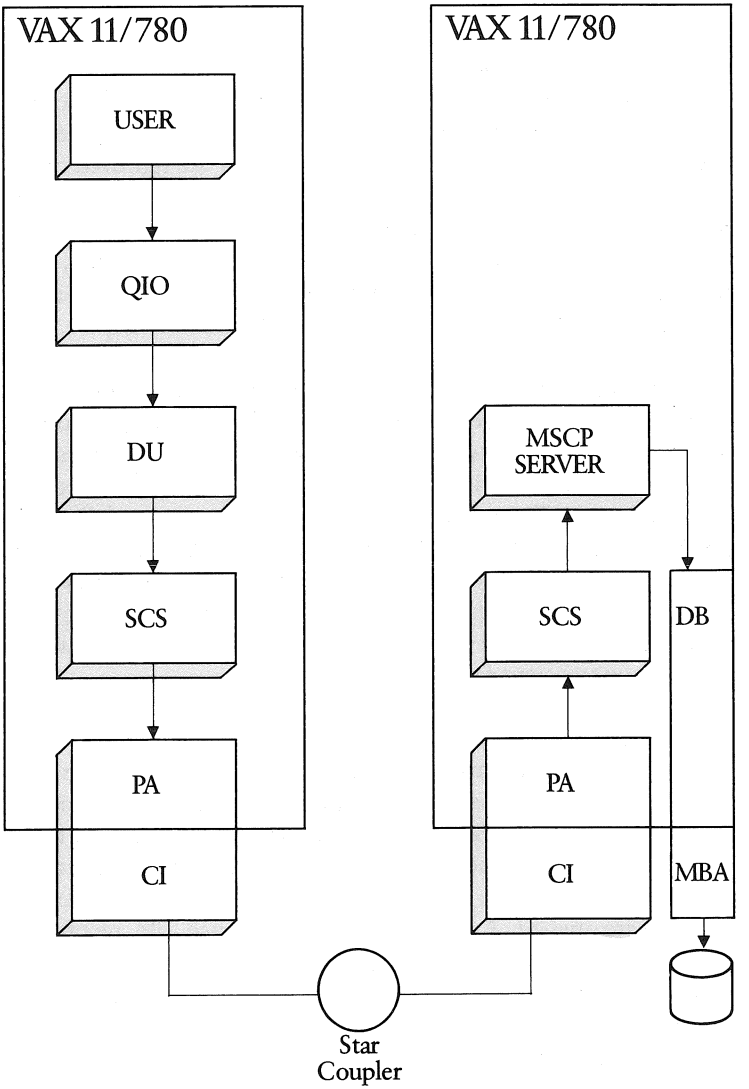
SCS provides a software interface to the three different transfer modes. An example is a disk I/O. An I/O request is made from a user's program. This is interpreted by the disk class driver, which produces the correct MSCP message. The MSCP message is then passed to the SCS level, which packages it as a sequenced message. The sequenced message is then transferred by the port driver through the CI port. As long as the disk to which the I/O was requested is on an HSC50, the HSC50 receives the MSCP message. After it has been decoded by the HSC50, the HSC50 then satisfies the I/O by transferring the data in a series of block data transfers, giving each one to the SCS level for correct packaging and transfer over the CI. When the I/O is completed, an MSCP message is sent via SCS from the HSC50 back over the CI to the disk class driver, thereby completing the I/O.

MSCP Server

Software called the MSCP server is responsible for one of the attractive features of the VAXcluster concept: preserving customers' investment in MASSBUS and UNIBUS disks. It does this by allowing MASSBUS and UNIBUS disks to be shared throughout the cluster.

MSCP is the protocol used to communicate between a VAX host and a DSA controller. If a user process running on VAX A in a VAXcluster has an I/O operation to perform to a MASSBUS or UNIBUS disk connected to VAX B in the same VAXcluster, the process queues its I/O to the disk class driver, as described above. The processing on the VAX node making the I/O request is the same, whether the I/O request is made to a MASSBUS or UNIBUS disk connected to another VAX processor or to a DSA device connected to an HSC50. The I/O request is transmitted in the form of an MSCP message communicated via SCS over the CI. When it arrives at the VAX processor (VAX B) that is connected to the MASSBUS or UNIBUS disk, the MSCP message is interpreted and translated into a MASSBUS or UNIBUS disk I/O request. This is the job of the MSCP server, which is also responsible for transferring the data over the CI.

The MSCP server enables a VAX processor to make locally connected MASSBUS and UNIBUS disks available to all other users of the VAXcluster. The system manager, however, decides which disks should be that widely available.



The I/O path of an MSCP-served disk.

Connection Manager

The chief function of the connection manager is to determine VAXcluster membership and to handle VAXcluster state changes. As new nodes are connected to the VAXcluster, or are removed or fail, the connection manager tracks the state changes and, in conjunction with connection managers running on each of the VAX processor nodes, ensures a consistent view of the cluster. The connection manager also prevents cluster partitioning via the QUORUM algorithm, which is described later in this chapter.

Another function of the connection manager is to provide an acknowledged-message delivery service. This means the connection manager is responsible for sending messages on the behalf of upper software levels via SCS and the CI to other VAX processors in the VAXcluster. The connection manager either successfully delivers the message or returns an error indicating that the target node has left or is about to leave the cluster. Transient failures are invisible to the upper layers of software. By adopting this approach, the higher-level system layers don't have to deal directly with SCS or transient failures.

Distributed Lock Manager

The distributed lock manager is a natural extension of the lock-management capabilities that were introduced for a single node in VMS Version 3. The distributed lock manager provides a name space for resource names; many types of locks may be taken out against the resources. Processes requiring access to a locked resource are queued. A resource could be a device, file, record in a file, or a fictitious resource used as a signaling mechanism by processes on different VAX nodes. The distributed lock manager allows synchronized use of resources throughout a VAXcluster. Automatic deadlock detection also has been expanded to work in a VAXcluster system.

The distributed lock manager is used by the file system, RMS software, job controller, device allocation, and other utilities for cluster synchronization and is available to users to develop cluster applications.

If a VAX node fails, all locks held by the failing node are released, thereby allowing the remaining VAX nodes to continue processing instead of stalling while waiting for a lock that may never be released.

The lock manager has been designed so that the cost of lock operations is fixed, regardless of the number of VAX processors in the cluster. As a result, the cost of synchronization is not affected by the size of the cluster. In other words, adding additional VAX processors does not in itself increase the synchronization cost.

Distributed File System and RMS

As with the distributed lock manager, the distributed file system is an extension of the file system that existed on VMS V3. The file system processes virtual I/O functions for mounted volumes. For disk volumes, this involves functions such as creating, deleting, and extending and truncating files, as well as maintaining file directories, mapping virtual to logical I/O, arbitrating runtime access to files, enforcing file protection, and enforcing and maintaining disk usage quotas. Most of these activities can interact with each other and therefore must be coordinated. Put another way, processing of various functions from different processes may have to be serialized.

In VMS V3, serialization was achieved by performing all of the above functions in the context of a detached process known as an ancillary control process (ACP). The ACP processes one request from start to finish before starting another, thus eliminating all possible interactions between different functions. Attempting to extend this design to a cluster would create a potentially serious bottleneck for virtual I/O activity. Instead, the VMS V4.0 distributed file system handles virtual I/O requests as an extended QIO procedure, or XQP.

XQP is a new technique developed to allow vastly greater concurrency for file system activity in the cluster. In addition, it benefits all VMS systems. The XQP uses the distributed lock manager to serialize virtual I/O functions for the specific file or directory in use. The code is mapped into the P1 area of each process. If the function modifies disk storage (for example, the creation, deletion, extension, or truncation of a file), the function is serialized with respect to other concurrent requests that modify disk storage on the same volume.

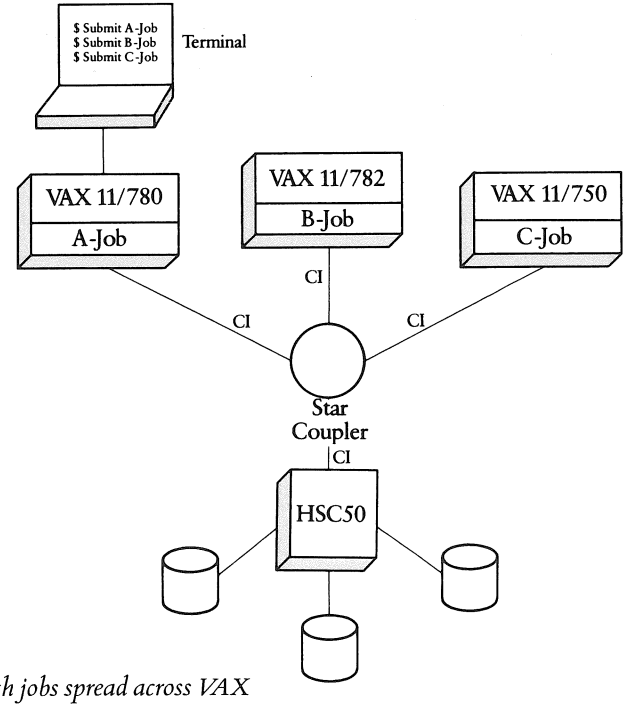
In practice, the XQP technique means that most file system functions may proceed in parallel with file system requests issued by other processes, whether the processes are executing on the same node, or on another node in the cluster. In addition, the various caches used in the VMS V3 ACP to reduce disk I/O have been reimplemented to perform similarly in the XQP environment. Various lock manager features are used to validate specific elements in the processor-specific caches within the cluster.

The VMS Record Management Services (RMS) have been extended on VMS V4.0 to take advantage of the VAXcluster environment. The RMS utility also has been interfaced to the distributed lock manager, allowing RMS files to be used on any disk in the cluster by any user on any VAX processor node. RMS files may be shared clusterwide at the record level, using standard RMS file-sharing and record-locking features.

Distributed Job Controller

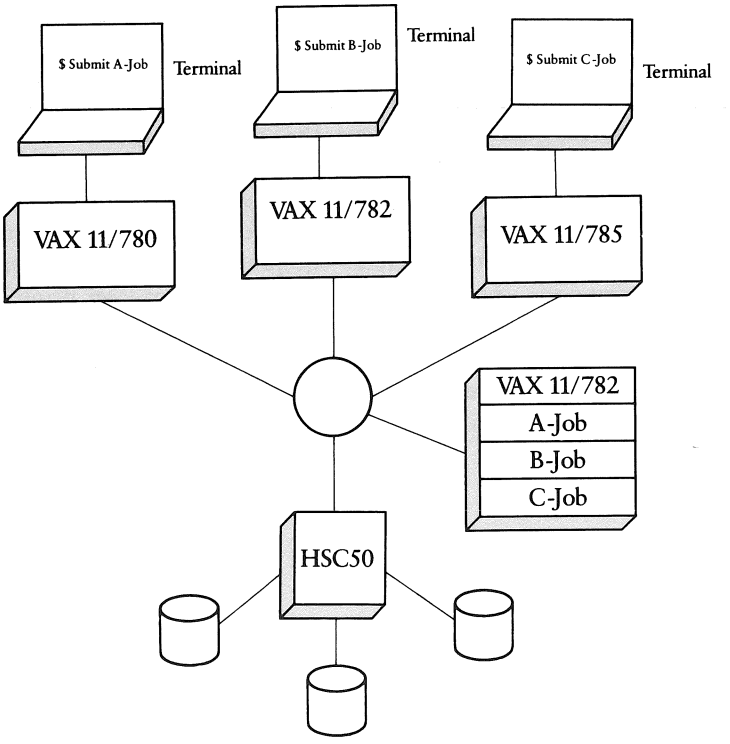
The distributed job controller allows the batch-and-print-queues database to be shared clusterwide. The distributed lock manager is used as a signaling mechanism to cause other VAX processor nodes in the cluster to examine the batch and print queues for jobs to be processed. Consequently, it is possible to create generic clusterwide batch and print queues, as follows:

- Clusterwide Batch Queues. In the case of a clusterwide batch queue, all batch jobs submitted to this generic queue are spread across the cluster. The algorithm attempts to keep each VAX processor equally loaded.



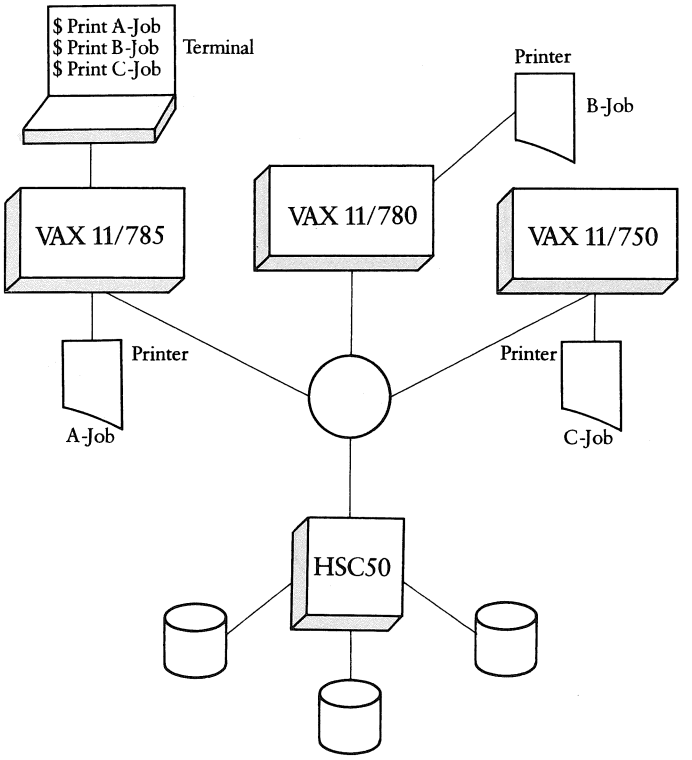
Batch jobs spread across VAX nodes.

A system manager can control how batch jobs are distributed as described above, where each node is a peer and all batch jobs are evenly distributed in number across the cluster. For example, if a VAXcluster consists of five VAX nodes and a user on the first VAX processor submits five batch jobs, one of the batch jobs may be run on each of the VAX processor nodes in parallel. At the other extreme, just one of the VAX processor nodes in the VAXcluster is the "batch" machine; in this case all batch jobs will run on the "batch" machine, regardless of the VAX node on which they were submitted. Those are the two extremes available to a system manager. But an intermediate situation is also possible. The load-balancing algorithm looks at the number of available batch slots and running jobs on each machine and attempts to keep their ratio even on each VAX node. Therefore, certain machines can be set to accept more batch work than others by setting up the batch queue parameters accordingly.

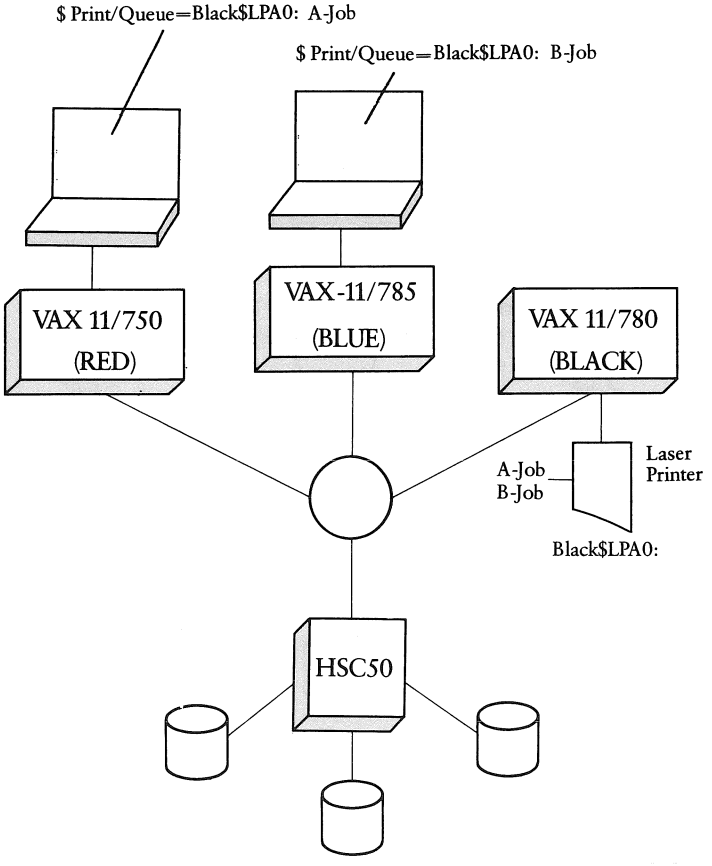


Configuration including a "batch" machine.

- Clusterwide Print Queues. It is also possible to have a generic clusterwide print queue. Assuming again that you have a cluster of five VAX nodes, each with a printer, if print jobs are submitted to the clusterwide generic print queue, the job will be directed to the printer device with the shortest queue. This allows print jobs to be spread across the VAXcluster. However, users may also submit their print jobs to any printer connected in the cluster. This is particularly useful when a cluster includes equipment that is not duplicated on every node, such as a letter-quality printer or laser printer. Print jobs may be submitted to this printer from any VAXcluster processor node. Or a cluster may have only one printer connected to one of the VAX processors; this printer, too, may be used from any of the VAX processors in the cluster.



Print jobs spread across VAX nodes.



Print jobs can be directed to a specific printer on any node.

Cluster Server Process

The cluster server process exists on each VAX processor node in the cluster. It performs global cluster functions. For example, it is used by the $Brdcst system service to broadcast messages to any or all terminals in the cluster. It is also used by the $Mount command to mount a disk globally on each of the VAX nodes. (The mount command need be issued only once from one of the VAX processors.)

DECnet in the VAXcluster

DECnet software is required in a VAXcluster for two reasons. First, a system manager will want to be able to control the cluster from any terminal and therefore may require the virtual-terminal capability that DECnet provides. Secondly, if communication is desired between processes on different processor nodes in the cluster, it is achieved with DECnet. Because the DECnet package has its own class driver, DECnet can utilize the CI directly, using the datagram service provided by the CI hardware. Each VAX processor in a cluster is seen as an individual DECnet node with a unique node name and node number. Digital requires that the DECnet node name and number be the same as the cluster node name and cluster identification number on each VAX processor.

Digital also recommends that every DECnet node in a VAXcluster be a full-routing node when the CI is being used. DECnet connections utilizing the CI are point-to-point; therefore, to prevent one node from routing all DECnet traffic, each node should be a full-routing node. VAX nodes could also be connected to Ethernet, which does not require the nodes to be full-routing nodes but allows them to be end nodes only. This permits the option of DECnet using the Ethernet connection to connect to other VAXclusters or DECnet

nodes in a local area network (LAN). The Terminal Servers described in the hardware section can also be placed on the LAN. Of course, each VAX processor node in the VAXcluster can have traditional point-to-point connections for long-distance wide-area connections or can use X.25 connections.

VMS V4.0 QUORUM Algorithm

A VAXcluster can share disks and their data. Consequently, the coordination of access to the data resources and the result it has on data integrity is very important. This is achieved in the cluster by each VAX processor node having a strong sense of cluster membership, which prevents two clusters from sharing the same data resource. A VAX processor node may only participate in one cluster. If sharing were allowed, between two or more clusters, the data resource could be written to and read in an uncoordinated manner, and data integrity could be lost.

The situation where two clusters share the same data resource in an uncoordinated manner is known as a partitioned cluster. Partitioning occurs when two or more nodes operate as if they belong to separate clusters although they are intended to be part of the same cluster.

With VMS V4.0, partitioning is prevented by a scheme known as the QUORUM algorithm. Each VAX node in the cluster has a SYSGEN parameter known as VOTES and a parameter known as QUORUM. Typically, each VAX node sets its VOTES parameter to 1 and the QUORUM parameter to the sum of VAX nodes divided by 2, plus 1, rounded down to the next integer value. (In this calculation, HSC50 devices are not counted as nodes.) For example, a cluster consisting of five VAX nodes and two HSC50 nodes would have a QUORUM of 3.

Processing may proceed only if the sum of the VOTES of the processors is at least equal to the QUORUM. In this example, if the cluster were to be partitioned into two clusters of three VAX nodes and two VAX processor nodes, then the three-node cluster would continue to function because it would still have 3 VOTES, enough to meet QUORUM. However, the second cluster would cease to function, since it has only 2 VOTES, or below QUORUM. Using this scheme, it is impossible for a properly configured cluster to become partitioned.

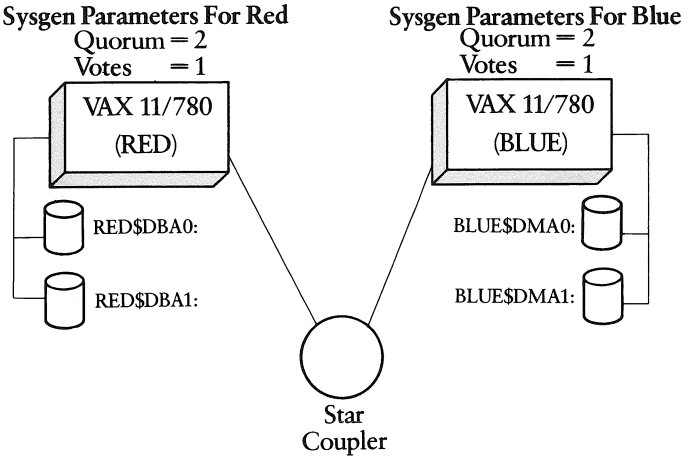
The QUORUM algorithm also affects the number of VAX processor nodes that may fail or leave the cluster before the remaining VAX processor nodes are affected. In the case of five VAX nodes with a QUORUM of 3, two VAX nodes may fail without affecting the remaining nodes. However, on the failure of a third node, the total number of votes falls below the QUORUM parameter and all processing in the cluster will stop until QUORUM is regained. If, however, after the first two nodes fail, it is known that the VAX processors are going to be down for some time, then the QUORUM value can be adjusted on each of the remaining VAX nodes by an operator or system manager. This means, for the example, that after the two nodes fail, there would be three remaining nodes with a QUORUM of 3. Consequently, the cluster could not tolerate a further failure because it would reduce the total number of VOTES below the QUORUM value. However, if the QUORUM value were adjusted to 2, then another VAX processor failure could be tolerated and the remaining nodes could continue running.

Therefore, it is apparent that the QUORUM algorithm has an effect on the configuration of VAXclusters. The bigger the cluster, the greater the number of VAX processor nodes that can fail before the number of VOTES drops below QUORUM and all processing is suspended in the cluster. The QUORUM algorithm works well for three or more VAX nodes but causes a problem in the case of two VAX nodes. With a cluster of only two VAX nodes, the QUORUM for the cluster is 2. Then, if either processor fails, the remaining processor would wait for QUORUM to be restored. This may be satisfactory for some VAXcluster users.

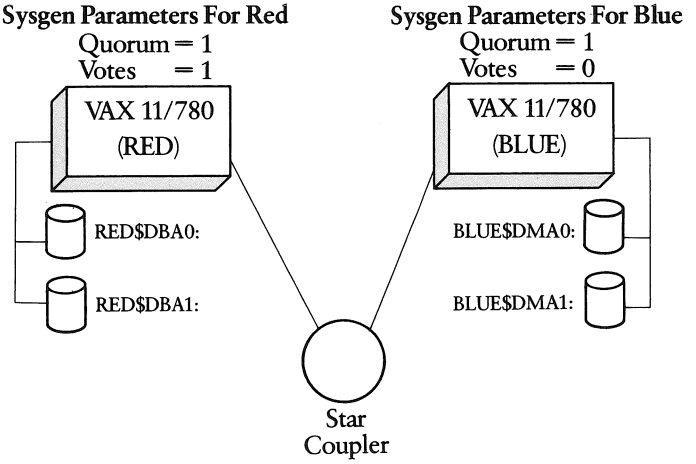
Another possibility is that a master-slave relationship is established. This can be achieved by giving the master processor 1 VOTE and the slave processor 0 VOTES, assuming a QUORUM of 1. In this case, if the slave VAX processor node were to fail, the master VAX processor node would continue to operate normally. However, if the master processor node were to fail, then both nodes would be unavailable until QUORUM were regained.

The third solution introduces one more concept to the QUORUM algorithm and is known as a quorum disk. A quorum disk is a disk that can be written to and read by all VAX nodes in a cluster. A quorum disk may be a disk on an HSC50 or for a two node cluster a dual-ported MASSBUS disk. The quorum disk is assigned a number of votes. If a cluster contains a quorum disk and it meets the criteria of being able to be written to and read by all VAX nodes, then the votes assigned to the quorum disk are counted towards the cluster QUORUM.

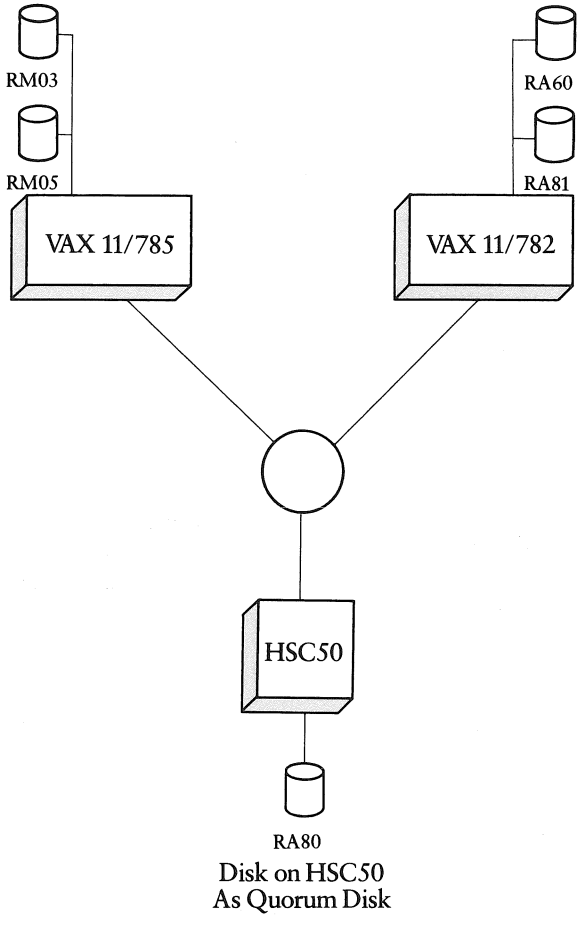
Therefore, in the case of a two-VAX-node cluster with a quorum of 2, if a quorum disk assigns one vote, then the failure of one of the processor nodes can be tolerated because there is still a quorum of 2 VOTES. In a normal running situation—both VAX nodes operating with a quorum disk—3 VOTES would be counted.



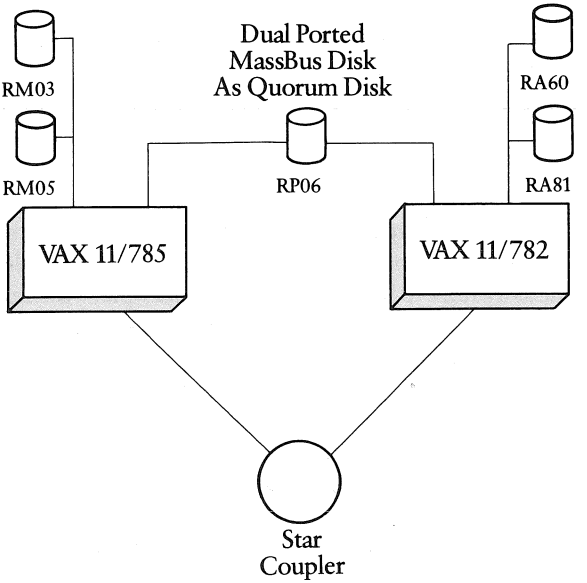
A two-node cluster with MSCP-served disks that cannot tolerate a processor failure.



The master-slave configuration.



A two-node cluster with an HSC disk as the quorum disk.



A two-node cluster with a dual-ported MASSBUS disk as the quorum disk.

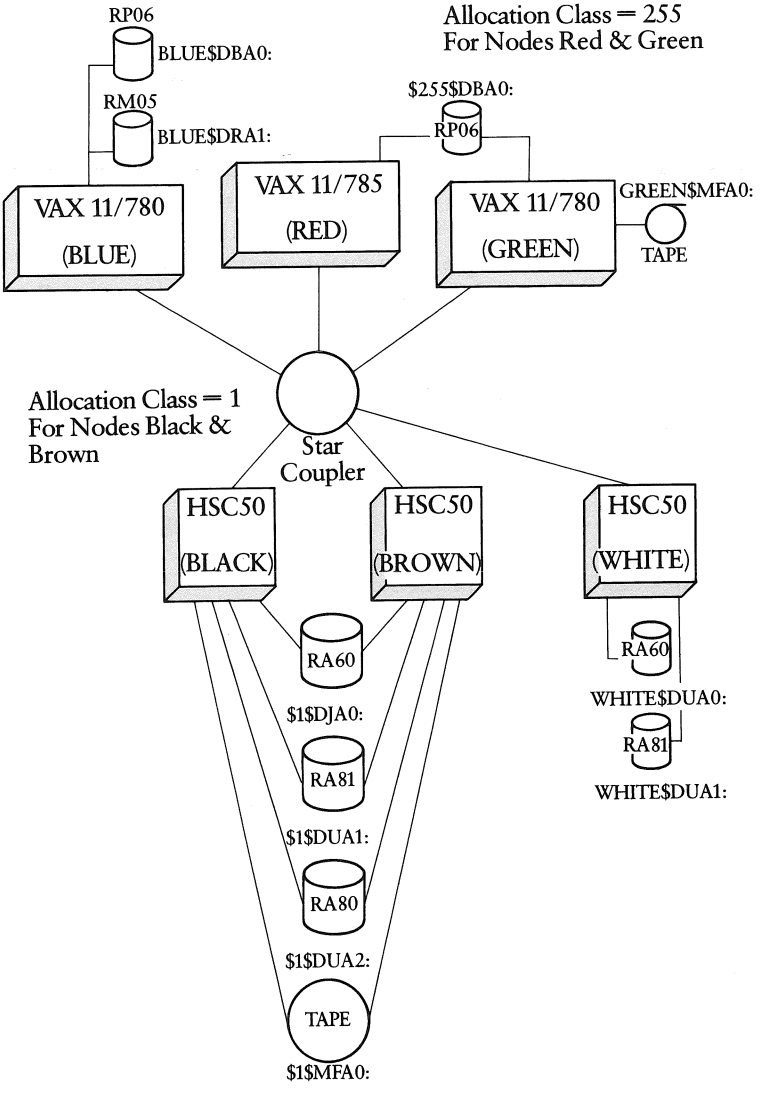
A quorum disk may be used by a cluster of any size to introduce extra VOTES, thereby tolerating the failure of extra VAX nodes.

Device-naming in a VAXcluster

Each node in a VAXcluster must have a unique name and SCS node number. To avoid confusion, Digital requires that the name and number be the same as the name and address of the DECnet node. The names consist of six characters and are assigned both to VAX processor nodes and to HSC50 nodes.

Each device in a cluster must also be identified by a unique name. A device connected to a particular node therefore would adopt the node name as a prefix to its own name. For example, a terminal line on a VAX processor named STAR might be STAR$TTA5: or an RA81 disk on a HSC50 with a name of PRIDE might be PRIDE$DUA0:. (RA80s and RA81s have the designation DU, and RA60s have the designation DJ.) Another example would be a MASSBUS disk connected to a VAX processor that is made available to the rest of the cluster via the MSCP server. An RP06 device connected to the VAX processor STAR could have the device name STAR$DBA0:. This approach varies only in the case of dual-ported disks. If disks are dual-ported between two HSC50s, or a MASSBUS disk is dual-ported between two VAX processors, the VMS operating system needs a universal device name that is known and doesn't change, regardless of what paths are available.

The answer is a concept called an allocation class. Each pair of VAX processors or HSC50s attached to a dual-ported disk are assigned a unique allocation-class number ranging from 1 to 255. For example, if a disk is dual-ported between the HSC50s named PRIDE and GREED, a unique allocation-class number, perhaps 255, must be assigned to both HSC50s. An RA81 disk dual-ported between the two HSC50s therefore might have a device designation 255DUA0:. The name is valid as long as either path is known.



A VAXcluster with dual-ported disks labeled with the correct allocation class.

Each disk connected to both VAX processors within the same allocation class must have a unique device name. For example, take the case of two VAX processors each having an RP06 device designated DBA0: and also sharing a dual-ported MASSBUS disk, thereby being assigned the same allocation class. In that case, one of the disks named DBA0: would be required to change its unit number to make it unique. The dual-ported MASSBUS disk must have the same name on both systems; that is, the same controller letter and the same unit number.

VMS V4.0 Features Not Supported Across the VAXcluster. Not all features available on a single VAX processor node are available across the cluster. Some features do not lend themselves to VAXcluster use. Other features not currently available may well be implemented in future stages of the VAXcluster program.

Features that are not implemented include

- Process control system services ($CREPRC, $SUSPND, $SETPRI)
- Accounting data
- Hardware error log
- Event flags
- Logical names
- Mailboxes
- Writable global sections

Time is maintained independently by each node. However, when a new cluster node boots, an attempt is made to set time from another node that is already in the cluster.

All of the above features continue to work on each of the cluster nodes. But, for example, an event flag set on VAX processor A in a cluster would not be set on VAX processor B of the same cluster.

It also should be noted that not every application is suited to VAXclusters. Care must be taken when migrating time-dependent applications to VAXclusters that require realtime, time-critical responses. The coordination and communication activity among VAXcluster members may require the suspension of processing activity for periods of time which, while relatively short, would not meet critical response times demanded by some applications.

Chapter 6· MANAGING A VAXcluster

A system manager has a number of choices when setting up a VAXcluster system. Each processor may have a separate User Authorization File (UAF) or there may be one UAF for the entire cluster. The UAF determines which users may log onto the system and, when a user has logged on, defines the user's default disk, directory, user identification code and privileges, and other entities. If each system has its own UAF, then different users may be restricted to certain VAX processors. However, the system manager must coordinate the different UAFs on the different VAX processors. For example, the system manager must ensure that a user has the same UIC on every VAX processor that is available to the user and that no two users have the same UIC.

The alternative is to have a single UAF placed on a shared disk for the entire cluster. In that case, the same UAF will be accessed whenever a user logs onto any VAX processor in the cluster. With this method, the user's process always will be defined the same way, making it irrelevant to a user which VAX processor he actually is using. If the cluster is set up this way and is connected to Ethernet, then the load-balancing feature of the Terminal Server may be used, as described in Chapter 3.

Single or Multiple System Disks

VMS V4.0 supports two or more VAX processors booted from the same physical disk drive. VMS V3 introduced the concept of rooted directories, which allow two or more VMS systems to reside on one disk, each under its own root named SYS0 through SYSF. VMS V4.0 uses this feature to allow multiple VAX processors to boot from the same disk drive. VMS V4.0 allows the VAX nodes of a VAXcluster to boot from any combination of the following:

- One VAX processor per system disk
- Multiple VAX processors per system disk sharing no system files
- Multiple VAX processors per system disk sharing virtually all system files

VMS V4.0 also will allow the sharing of VMS and most layered products. Portions of those products that must be separate for use by each VAX processor are placed in a separate rooted directory. Therefore, the majority of VMS and layered products are placed in a common directory to be shared by all nodes booting from the disk. This greatly decreases the complexity of managing VMS and layered products in a cluster, since any ongoing software updates have to be applied only once.

Plans to use shared system disks prompts the question of how many VAX processors should be booted from the same disk. The answer almost certainly will be determined by performance. If the shared VMS and layered-product option described above is implemented, then all of a system's software and layered products can be considered in three parts. They are the portions of the system that must reside in a unique rooted directory for each VAX processor, the common directory containing common VMS and layered-product components, and each VAX processor's page- and swapfiles.

How these sections of the system are spread over cluster disks will determine the system's ultimate performance. The decision should be based on performance rather than on space requirements. It also should be remembered that, with multiple VAX processors using one disk, it is relatively easy to reach the I/O capacity of the drive, resulting in a system bottleneck. Further more, should a system disk fail, all systems booting from it fail.

DECnet

With VMS V4.0, DECnet software must be available in a VAXcluster. This is convenient for the system manager because the cluster can be controlled from one terminal, thanks to the virtual terminal capability that DECnet provides. Each cluster node name and number must be the same as the DECnet node name and number. Each DECnet cluster node also should be a full-routing node because DECnet utilizes the CI as a point-to-point configuration.

System Management Tools:

Show Cluster Utility

The Show Cluster utility is new with VMS v4.0. It produces a continually updated, user-specific display of different aspects of the cluster's operation. This allows a system manager to monitor the overall operation of the cluster.

CLUSTER					
CL-Q	CL-V	QD-NAME	QF-V	FORMED	LAST TRANSITION
3	4	255DUA5:	YES	15-AUG-84 08:08	20-AUG-84 09:02

SYSTEMS		MEMBERS	CIRCUIT	COUNTERS			
NODE	SOFTWARE	STATUS	CABLE	DGS-S	DGS-R	MSGS-S	MSGS-R
Galaxy	VMS V4.0	Member	A-B	141	428	148	148
Blue	VMS V4.0		A-B	0	0	3019	4299
Star	VMS V4.0	Member	A-B	0	0	3340	4608
Helos	VMS V4.0	Member	A-B	0	0	1120	1120
Meteor	VMS V4.0	Member	A-B	0	0	592	592
Deceit	HSC V110		A-B	0	0	42	42

A possible show cluster screen.

Monitor Utility

VMS V4.0 continues to enhance and add capabilities to the Monitor utility. Features that are new with VMS V4.0 are displays for monitoring the lock manager, SCS, and disk I/O on a per-disk basis. The Monitor utility can collect data on different VAXcluster nodes and play the data back, side by side on the same screen. This can be used to help balance the load between VAX processors and to identify busy disks that need to be offloaded.

User Environment Test Package

The User Environment Test Package, or UETP, has been enhanced to include cluster verification tests. It ensures that locks taken out on nodes are visible and that cluster deadlocks will be detected. For every node with cluster disks, UETP creates and updates a file and checks whether it can be shared by member nodes. UETP usually is run at installation time.

Cluster Operations

All operator messages generated by the system or by users are dispatched to all operator terminals in the cluster, regardless of the VAX processor to which the operator terminal is connected.

Installation of a VAXcluster:

What's Involved?

This section is an overview of how to install a VAXcluster system. Details may be found in the VMS V4.0 documentation. Digital recommends that each VAX processor node be set up individually. First, install VMS V4.0 and then DECnet on each of the VAX processor nodes. Second, boot each VAX processor node separately and run it in a single-node environment. When satisfied that everything is set up correctly on the VAX node, close it down and move on to the next one.

Once all the VAX processor nodes have been set up individually, they may be booted together. At that point, a VAXcluster will be formed. The system manager should then adjust the QUORUM parameter to the appropriate value for the cluster while booting the cluster.

Upgrading A Running VAXcluster

An HSC50 or VAX processor may be added to a running VAXcluster system without interrupting the cluster.

A DSA disk or tape also may be added to an HSC50 while the cluster is running, assuming that enough SDI or STI interfaces are available. When the HSC50 detects a new device, it informs the VAX processors in the cluster that a new device is available. The VMS operating system on each VAX processor then dynamically adds to its I/O database. The device may then be mounted.

More HSC50s may be added to a VAXcluster. The CI cables are attached to the Star Coupler, as long as enough connectors are available in the Star Coupler. The Star Coupler has an 8- and 16-node configuration. Adding another HSC50 or VAX processor may require upgrading the Star Coupler to a 16-node configuration. With adequate connections, once the HSC50 is added and booted, each VAX processor can detect the new HSC50 and any disks connected to it. The cluster will continue to function normally.

Chapter 7 · VAXcluster FUTURE DIRECTIONS

The VAXcluster program was announced in May 1983. VMS V3.3 and subsequent releases carried out the first phase of the cluster program. VMS V4.0 brings many new and exciting capabilities to the VAXcluster approach and may be considered the second phase of the VAXcluster program. In the future the final pieces of the announced program will be added. However, the completion of the VAXcluster program as announced in May 1983 is just the beginning of capabilities that will take advantage of the cluster approach.

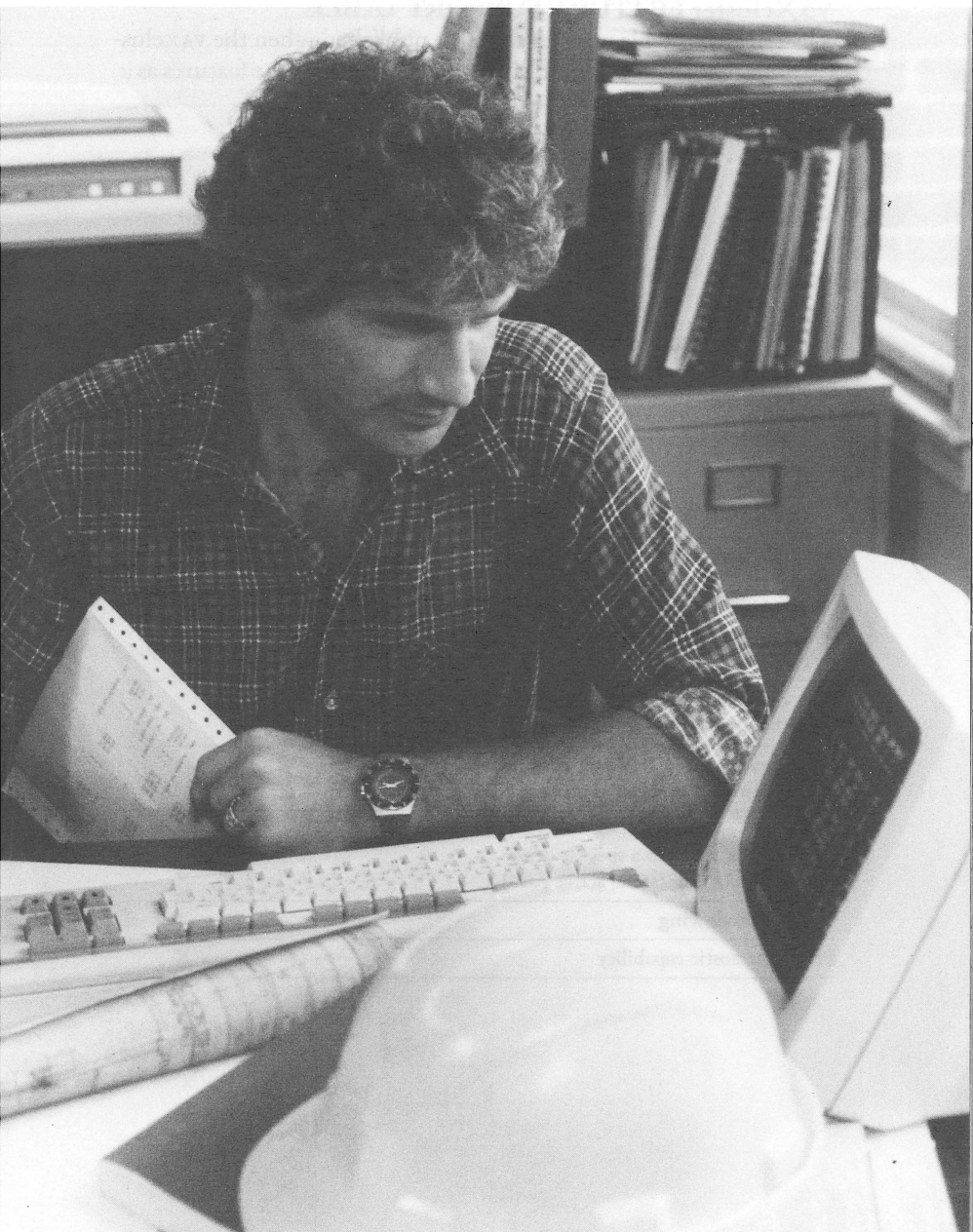
In the future, more capabilities will be added to the VAXcluster environment. These include volume shadowing, common journaling, and checkpoint/restart.

- Volume shadowing allows the system to tolerate disk failure in what is sometimes called a "hot standby" mode. This capability will be provided by the HSC50 and by the MSCP Server for disks on the UDA50.

Volume shadowing defines a virtual disk unit that represents a collection of disks known as the shadow volume set. Data written to the shadow volume set is written to all the disks in the set. Data read from the shadow volume set is taken from any one of the members of the set. If the shadow set is primarily read from, performance may improve. This can occur because more than one drive is available to be read from, thereby satisfying read requests faster. If one of the disks in the shadow set fails, it will be failed out of the shadow set and processing will continue. The failover to the remaining disks in the shadow set is transparent to users, allowing the disabled disk to be removed and repaired. Once it is operational again, it may be placed back into the shadow set; "catch-up" will be performed on the drive to make its contents identical to that of the other drives.

- The *common journaling* facility will provide a generic journaling capability on the VMS operating system. VAX DBMS and VAX RDB/VMS now provide journaling as part of their product on VMS, and in the future, a common journaling capability will be added to RMS. This will be implemented to provide data integrity, both on single VAX processors and in a VAXcluster.
- The *checkpoint/restart* feature will allow long-running jobs to be checkpointed periodically. If the job fails, or the VAX processor on which the job is running fails, the job could be restarted from the previous checkpoint. In a cluster, jobs may be restarted on any CPU identical to the failed CPU.

APPENDIX I



VAXcluster FEATURE SUPPORT TABLE

The following table updates the table published when the VAXcluster program was announced. It shows the status of the features as a result of VMS V4.0.

Table: VAXcluster Feature Support

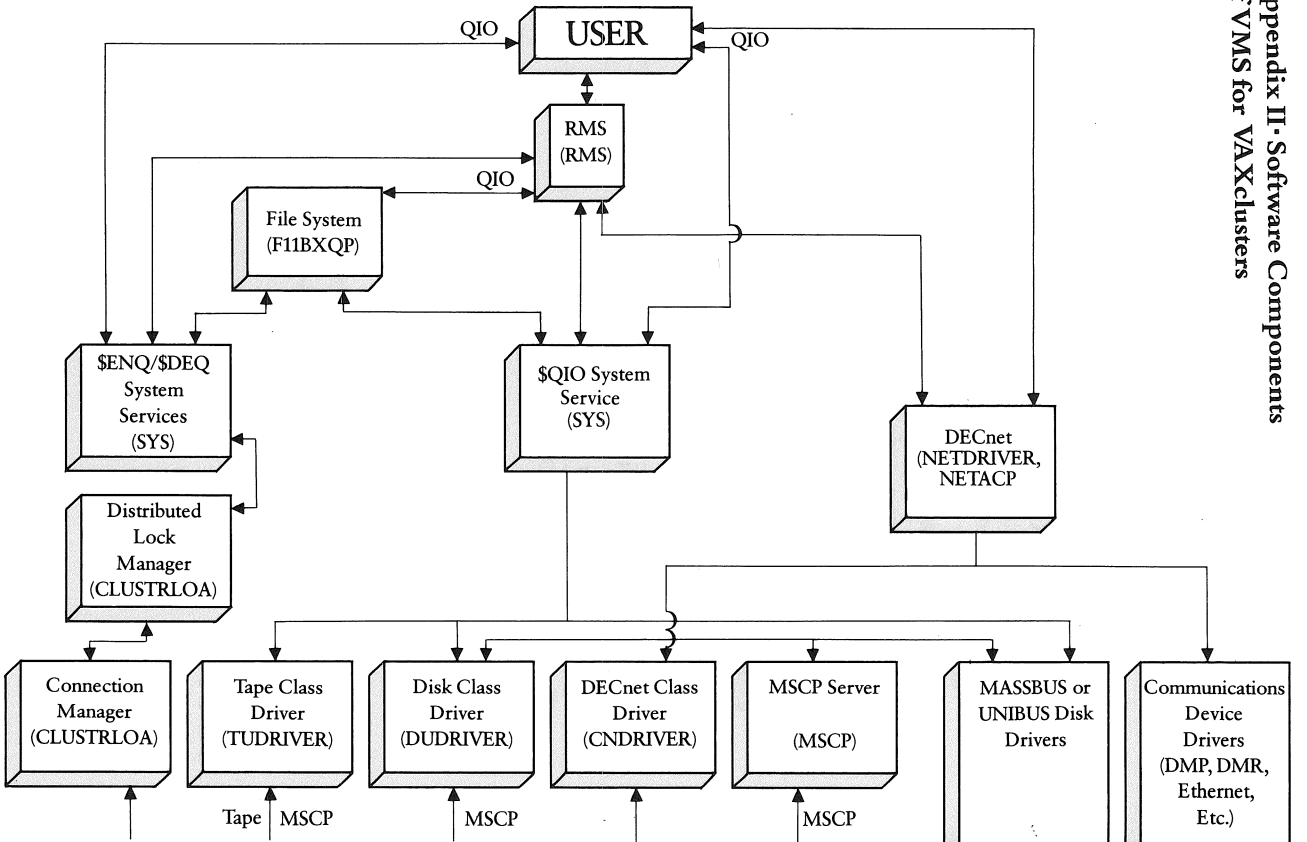
Key to table:

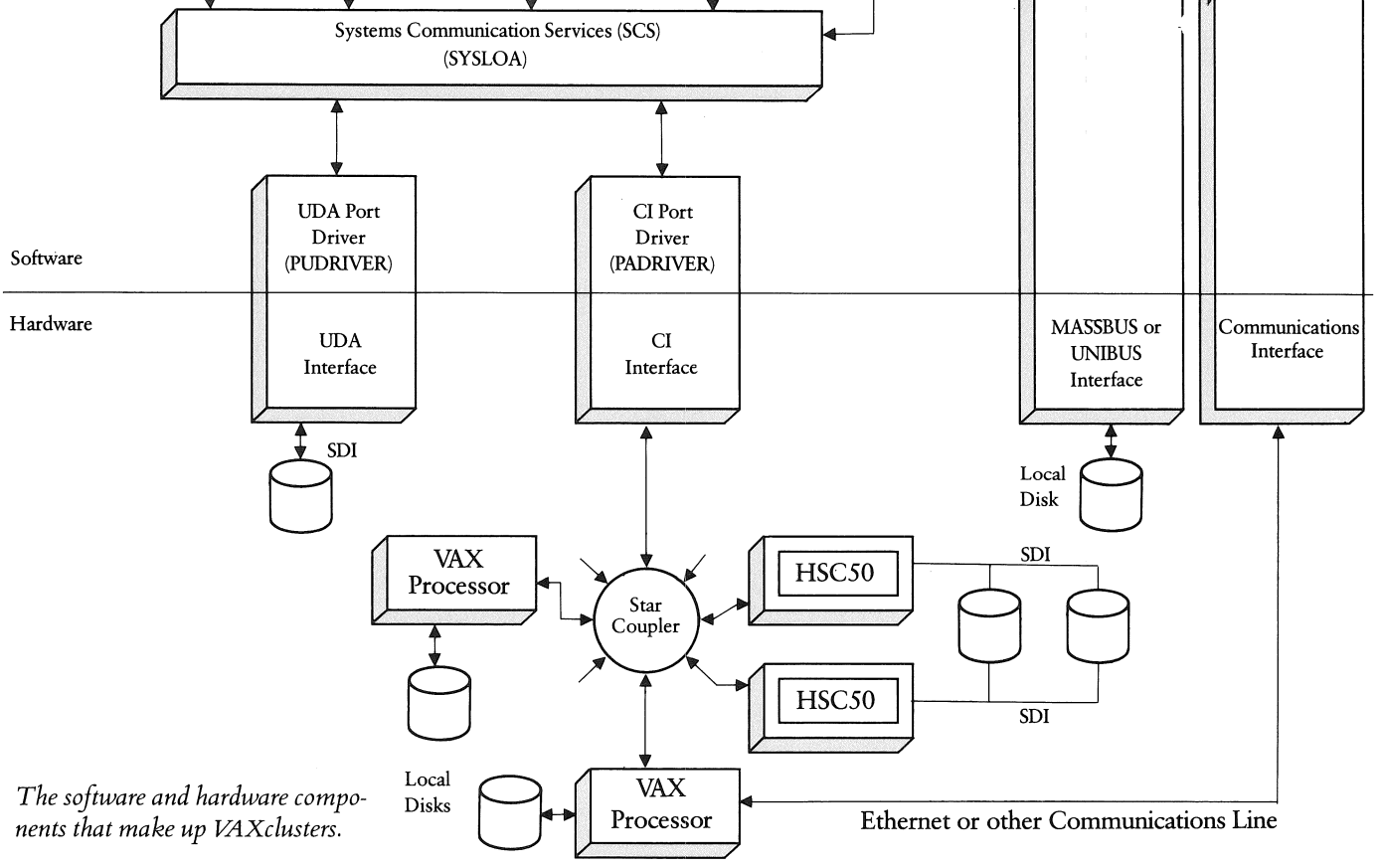
X Facility introduction
* Restriction removed

FEATURE	VMS V3	VMS V4.0	FUTURE
CI—VAX-11/780, VAX-11/782, DECnet-VAX support	X		
CI—VAX-11/750, VAX-11/780, VAX-11/782, VAX-11/785, DECnet-VAX support	X		
HSC50 Support	X		
Disks	X		
A CPU has access to a specific volume	X	*	
Volume sharing (one writer, multiple readers)	X	*	
Seek optimization	X		
Rotational position optimization	X		
Disk dual-porting (static active path)	X		
Auto failover (static active path)		X	
Tapes			X
A CPU has access to a specific drive		X	
Tape dual-porting (static active path)			X
Volume shadowing			X
Host diagnostic capability			X

FEATURE	VMS V3	VMS V4.0	FUTURE
VAX-11/780s, VAX-11/782s, and HSC50s on the CI	X		
Each CPU requires own system disk (local or HSC50)	X	*	
No local mass storage required	X		
VAX-11/750s become part of the VAXcluster	X		
Local disk mass storage required (for booting)	X	*	
Data Integrity Facilities			X
Common Journaling Facility			X
Recovery Unit Facility			X
VAX-11 RMS Journaling and Recovery			X
Checkpoint/Restart Facility			X
Distributed Lock Manager		X	
Clusterwide Transparent Disk Access (Local or HSC)		X	
Distributed File System		X	
File/Record-level sharing on all disks		X	
MSCP Server		X	
Clusterwide access to local disk storage		X	
Clusterwide Batch/Print Queues		X	

Cluster Hardware/Software Block Diagram





The software and hardware components that make up VAXclusters.

Notes

Notes

Notes

digital

ORDER CODE: ED 26065-98