T W E N T Y - F I V E

Setting Up a Serviceguard Cluster

▲Chapter Syllabus

This is where we get down to the *nitty-gritty* of setting up a Serviceguard cluster on HP-UX. We mention concepts such as "*mirroring your root disk*." The mechanics of performing such tasks are not covered in this module because they are covered elsewhere. Specific tasks that are relevant for setting up a Serviceguard high availability cluster are covered and, where appropriate, screenshots and example commands are used to help illustrate key points.

25.1 The Cookbook for Setting Up a Serviceguard Package-less Cluster

Before we can start to use this cookbook, we need to understand what Serviceguard is and what it is trying to achieve. I suggest that you *not jump* straight to the *cookbook* (see Table 25-1) because having an understanding of the concepts and limitations of Serviceguard can influence your decisions on how to construct your cluster. Each bullet point in the cookbook should be studied, understood, and implemented carefully. So here it is.

Table 25–1 *Cookbook for Setting Up a Serviceguard Package-less Cluster*

Cookbook for Setting Up a Serviceguard Package-less Cluster:
1. Understand the hardware and software implications of setting up a cluster.
2. Set up NTP between all cluster members.
3. Ensure that any shared LVM volume groups are not activated at boot time.
4. Install Serviceguard and any related Serviceguard patches.
5. Install a Quorum Server (optional in a basic cluster).
6. Enable remote access to all nodes in the cluster.
7. Create a default ASCII cluster configuration file (`cmquerycl`).
8. Update the ASCII cluster configuration file.
9. Check the updated ASCII cluster configuration file (`cmcheckconf`).
10. Compile and distribute the binary cluster configuration file (`cmapplyconf`).
11. Back up LVM structures of any cluster lock volume groups (`vgcfgbackup`).
12. Start cluster services (`cmruncl`).
13. Test cluster functionality.

Before we get started, we need to begin by talking about the unthinkable—a failure. What constitutes a failure? Different types of failure will prompt different responses from Serviceguard. This is where we start our discussion.

25.2 The Basics of a Failure

The users will connect to the application through an application or **application package** IP address, thus, removing the dependency between an individual server and an individual application. In the event of a "failure," the application will be restarted on another node that we will refer to as an "*adoptive node*." Essentially, we want our applications to run on a pre-

scribed machine for as long as possible and, hence, eliminate the necessity to restart it on an adoptive node. Here is a list of the general points of what constitutes a failure:

- **A failure of all LAN communications:** If we had a Standby LAN card, Serviceguard would use it. Otherwise, the application package will be moved to an adoptive node.
- **Total system failure:** The cluster will detect a node is no longer functioning and restart an application package on an adoptive node.
- **Application failure:** The cluster is monitoring prescribed application processes. If such a process dies, Serviceguard has two option: restart the process a prescribed number of times, or restart the application on an adoptive node.
- **Other critical resources fail:** Serviceguard can be configured to utilize the Event Monitoring Service (EMS) that can monitor the state of critical system components and resources. Should one of these components or resources fail, an application package will be moved to an adoptive node.

25.3 The Basics of a Cluster

A cluster is a collection of at least two nodes and up to 16 nodes. Supported cluster configurations include:

- **Active/Active:** This is where all nodes are running their own application package but can run additional application packages if necessary.
- **Active/Standby:** This is where a single node is not actively running any application packages but is waiting for a failure to occur on any of the other nodes in the cluster, whereby it will adopt responsibility for running that nodes application package.
- **Rolling Standby:** This is similar to Active/Standby in that we have a node that is waiting for a failure to occur on any node in the cluster. The difference here is that when a failure occurs, the *failed* node becomes the *standby* node after the initial problem is resolved. Should a second failure occur, the second *failed* node becomes the *standby*. In a purely **Active/Standby** configuration, if a second failure occurred, the original *standby* node would be running two application packages.

Cluster monitoring is performed by a number of Serviceguard processes. Serviceguard has three main management functions:

- **Cluster Manager:** The management and coordination of cluster membership.
- **Network Manager:** Monitoring network connectivity and activating standby LAN cards when necessary.
- **Package Manager:** The management of starting, stopping, and relocating application packages within the cluster.

The main Cluster Management Daemon is a process called `cmcld`. This process is running on every node in the cluster, and one of its main duties is to send and receive *heartbeat* packets across all designated heartbeat networks. One node in the cluster will be elected the

cluster coordinator that is responsible for coordinating all inter-node communication. The **cluster coordinator** is elected at cluster startup time and during a cluster reformation. A cluster will reform due to one of four events:

- A node leaves the cluster, either "gracefully" or because the node fails
- A node joins the cluster
- Automatic cluster startup
- Manual cluster startup

When we set up our cluster, we discuss this "election" in a bit more detail. A critical feature of the cluster coordinator is the detection of a node failure; in this, we mean either total LAN communication failure or total system failure. For every `HEARTBEAT_INTERVAL`, nodes are transmitting a heartbeat packet on all prescribed heartbeat interfaces. If this heartbeat packet does not reach the cluster coordinator, after a `NODE_TIMEOUT` interval the node is determined to have failed and a cluster reformation commences. It goes without saying that maintaining heartbeat communication is vitally important to all nodes in the cluster.

- **Cluster HEARTBEAT and STANDBY LAN interfaces:** Because this is such a crucial part in determining the health of a cluster, the more LAN interfaces you prescribe as being heartbeat interfaces, the better. The only time this is not the case is if you are intending to use VERITAS Cluster Volume Manager (CVM). The design of CVM allows the CVM daemon `vxclustd` to communicate over a single IP subnet. You will realize when you try to run `cmcheckconf` and `cmapplyconf`. If more that one heartbeat LAN is configured in a CVM configuration, both commands will fail. LAN interfaces can either be designated as a `HEARTBEAT_IP` (carries the cluster heartbeat) or a `STATIONARY_IP` (does not carry the cluster heartbeat). Even if a LAN interface is configured as a `HEARTBEAT_IP`, it can carry normal application data as well. The designation `STATIONARY_IP` simply means that no heartbeat packets are transmitted over that interface; it *does not* mean the IP address cannot be moved to a redundant, standby LAN card. The use of a redundant standby LAN interface for all interfaces is highly recommended. If you are going to use only one standby LAN card for all LAN interfaces, it must be bridged to all the networks for which it is being a standby. Figure 25-1 shows a good setup where we have a standby LAN card for each active network.

In Figure 25-1, you will also notice that the `HEARTBEAT_IP` is not being utilized by any clients for data traffic. This is an ideal scenario because `HEARTBEAT` packets are not contending with data packets for access to the network. You can use a `HEARTBEAT_IP` for data traffic as well, although you should note that if data traffic becomes particularly heavy, then the heartbeat packet may not reach the cluster coordinator, and this could cause a cluster reformation because some nodes "appear" to have "disappeared." You should also note that the standby LAN cards are bridged with the active LAN card. This is absolutely crucial. Serviceguard will poll standby/active LAN cards every `NETWORK_POLLING_INTERVAL` to ensure that they can still communicate. The bridge/switch/hub that is used should support the 802.1

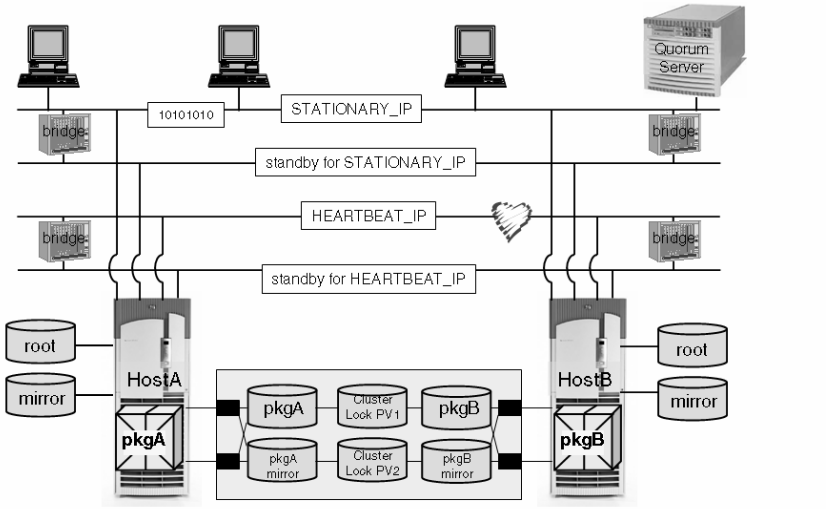

Figure 25–1 *The use of standby LAN cards.*

Spanning Tree Algorithm (most of them do). The Quorum Server is currently attached to the main corporate data LAN. This is not a requirement. It just shows that all nodes in the cluster must be able to communicate with it, and it could be "any" machine in your organization running HP-UX. Many customers I know have the Quorum Server attached to the dedicated Heartbeat network. I think this is a good idea because all nodes in the cluster need access to the Heartbeat network and when we need to communicate with the Quorum Server, we are not competing with other users for access to our corporate data LAN.

When I first looked at Serviceguard I wondered, "How many LAN cards do I need?" The simple answer is two. Serviceguard is designed to fit into the whole philosophy of high availability. If Serviceguard "*allowed*" you to run with just one LAN card, it would be an immediate SPOF. So you need two LAN cards, with one acting as a STANDBY. Well, if I am really honest, you can get away with one LAN card. In a simple two-node cluster similar to the one you can see in Figure 25-1, you could use only one LAN card as long as you used an RS-232 null modem cable as a "serial heartbeat" between the two nodes. The one LAN card needs to be used as a `HEARTBEAT_IP`, i.e., in this case for data plus heartbeat packets. The serial heartbeat will be used as a *last-ditch* means for nodes to communicate in the event of network saturation. In that instance, both nodes will use the serial link to determine who was the "*best candidate*" to reform the cluster on their own. (Note: The use of serial heartbeats is being viewed as "*less than ideal*" and may be phased out in the near future.) In essence, the serial heartbeat is adding a little intelligence into the cluster reformation process, but only when we have a two-node cluster with only one LAN card in each node. This leads me to my next point about High Availability Clusters: the "split-brain" syndrome.

25.4 The "Split-Brain" Syndrome

The "split-brain" syndrome can easily be described if we consider a simple two-node cluster like the one in Figure 25-1. If these nodes were to lose all LAN communications, how would they decide who was the "best" to reform the cluster? Even if they had a serial heartbeat, we could still be in a situation where both nodes had individual communications but for some reason could not communicate with each other. Serviceguard requires a *cluster quorum* of more than 50 percent of the previously running nodes. In the two-node situation described above, we could be in a situation where two equal-sized clusters would both try to reform, and if allowed to do so, we would have two instances of our applications running simultaneously—and that's not a good idea. In this situation, we need a "*tiebreaker*." For Serviceguard, the tiebreaker is known as a *cluster lock*. Serviceguard now offers two forms of *tiebreaker*:

- **Cluster Lock Disk:** This is a shared disk that both nodes can see and that is controlled by LVM. In a cluster of more than four nodes, a **cluster lock disk** is not supported or allowed. A **quorum server** is.
- **Quorum Server:** This is a separate node, not part of the cluster but contactable over a network interface (preferably on the same subnet to avoid delays over routers, and so on). This machine could be something as simple as a workstation running HP-UX 11.0 or 11i (either PA-RISC or IPF), or it could even be a machine running Linux. The Quorum Server listens to connection requests from the Serviceguard nodes on port #1238. The server maintains a special area in memory for each cluster, and when a node obtains the cluster lock, this area is marked so that other nodes will recognize the lock as "taken." It may provide quorum services for more than one cluster.

The idea of a **cluster lock** can be extended to a cluster of any size; we do not want two groups of nodes each containing 50 percent of the nodes previously in the cluster trying to form two clusters of their own. Again, we would have two sets of applications trying to start up simultaneously—not a good idea. We need a **cluster lock** in a two-node cluster; it's a must because of the "**split-brain**" syndrome. In a three-node cluster, it is advisable because one node may be down for maintenance and we are back to being a two-node cluster. For more than three nodes, a **cluster lock** is optional because the chance of having two groups of nodes of exactly equal size is unlikely. Whichever group of nodes wins the "tiebreaker," those nodes will form the cluster. The other group of nodes will shut down by instigating a TOC (Transfer of Control). We look at a crashdump later, which tells us that Serviceguard caused the system to initiate a Transfer Of Control (TOC).

Before we get started on actually configuring a cluster, I want to use just one last paragraph to remind you of some other hardware considerations.

25.5 Hardware and Software Considerations for Setting Up a Cluster

I won't go through every permutation of supported disk and LAN technologies. But I do want to jog your memory about Single Points Of Failure in relation to hardware components. I will leave it up to you to perform a hardware inventory to ensure that you do not have an SPOF in your design.

1. **SPU:** It is not a requirement for each node in a cluster to be configured exactly the same way, from a hardware perspective. It is not inconceivable to use a lower-powered development server as a Standby node in case your main application server fails. You should take some time to understand the performance and high availability implications of running user applications on a server with a dissimilar configuration.
2. **Disk Drives:**
   i. These are the devices that are most likely to fail. Ensure that you offer adequate protection for your operating system disks as well as your data disks.
   ii. Utilizing highly available RAID disk arrays improves your chances of not sustaining an outage due to a single disk failure.
   iii. If you are utilizing Fibre Channel, ensure that each node has two separate connections to your storage devices via two separate Fibre Channel switches.
   iv. Ensure that hardware solutions have multiple power supplies from different sources.
   v. Software components can offer RAID capabilities as well; LVM can offer RAID 0, 1, 0/1. VxVM can offer RAID 0, 1, 0/1, 1/0, and 5. When utilizing software RAID, ensure that mirrored disks are on separate controllers and powered from a separate power supply.
3. **Networks:**
   i. Ideally, have at least one separate heartbeat LAN.
   ii. Ideally, have a standby LAN for all LAN interfaces, including heartbeat LANs.
   iii. Utilize multiple bridges/switches between active and standby LANs.
   iv. If utilizing multiple routed IP networks between servers and clients, ensure that multiple routers are used.
      a. Ensure that all members of a network can support dynamic routing.
      b. Endeavor to utilize the most robust routing protocols, e.g., RIP2 or even better OSPF.
   v. If utilizing FDDI, ensure that dual attached stations are attached to separate concentrators.
4. **Power Supplies**
   i. Ensure that you have at least two independent power supplies.
   ii. Independent power supplies should be fed from two external power generators; that includes power supply companies. Take the situation where you have two inde-

pendent power feeds into your data center both from XYZ Generating Company. If XYZ Generating Company goes "bust" or loses the capability to supply you with electricity, you are somewhat in a pickle. If all else fails, your second power supply should come from an onsite generator.

iii. Regularly test the ability of your second power supply to "kick in" seamlessly when your primary supply fails.

5. **Data Center:**
   i. You need to consider your data center as an SPOF. How are you going to deal with this? This can involve a Disaster Recovery Plan including offsite tape data storage or could be as sophisticated as an advanced cluster solution such as Metrocluster or Continentalclusters incorporating asynchronous data replication over a DWDM or WAN link.
   ii. Ensure that management understands the implications of not including your data center in the overall High Availability Plan.
6. **Performance:**
   i. Should an application fail over to an active adoptive node, you will have two applications running on one node. Do the consumers of both applications understand and accept this?
   ii. Have you set up any Service Level Agreements with your user communities relating to individual application performance?
   iii. How will you manage the performance of individual applications in the event of a failover to an active adoptive node?
   iv. Will you employ technologies such as Process Resource Manager (PRM) and Work Load Manager (WLM), or leave performance management to the basic UNIX scheduler?
7. **User Access:**
   i. Do users need to perform a UNIX login to the node that is running their application?
   ii. Does the user's application require a UNIX user ID to perform its own level of client authentication?
   iii. How will you manage providing consistent UNIX user and group IDs across the entire cluster?
   iv. Do you use NIS, NIS+, or LDAP?
   v. Do you use Trusted Systems?
8. **Security:**
   i. Do you have a security policy for individual nodes?
   ii. Do you have a security policy for your network(s)?
   iii. Do you have a security policy for the cluster?
   iv. Do you have a security policy for your data center?
   v. Do you have a security policy for users?
   vi. Does everyone concerned know and understand your security policies?

vii. Do you employ third-party security consultants to perform penetration tests?

viii. Does your organization have an IT Security Department? If so, do they perform regular security audits? Do they understand the security implications of an HP High Availability Cluster?

That should jog your memory about the SPOF and matters relating to the availability of your applications. As you can see, it's not just about "*throwing*" hardware at the problem; there are lots of other technological and process related challenges that you will need to face if you are to offer maximized uptime to your customers.

Let's move on to look at the mechanics of setting up a basic High Availability Cluster.

25.6 Testing Critical Hardware before Setting Up a Cluster

Let's start by looking at some "*tips and tricks*" of setting up our key hardware components. We'll begin with disk drives:

- **Disk Drives:** There are two scenarios I want to consider here:
  - **Using VxVM disks:** When using VxVM disks, we should give each disk an easily identifiable Disk Media name. In this way, when we deport/import disk groups, we can identify disks easily; remember, it is possible that device file names will not be consistent across machines in the cluster, so Disk Media Names will be the identifier in this case.
  - **Using LVM disks:** Using LVM disks poses its own problems. The identifier for an LVM disk is the device file. We need to know the device file name to be able to import the volume group into all nodes in the cluster. Here's what I do:
    1. Set up the volume group on one node in the cluster. This includes all logical volumes and filesystems. Create all the mount points necessary.
    2. Load all data/files into the appropriate filesystems and logical volumes.
    3. Do *not* update the file `/etc/fstab`.
    4. Test that you can see and access all data/files appropriately.
    5. Create a map file (in preview mode) from the active volume group. Here is an example of the error message you will see:

```
root@hpeos001[] # vgexport -p -m /tmp/vg01.map /dev/vg01
vgexport: Volume group "/dev/vg01" is still active.
root@hpeos001[] # cat /tmp/vg01.map
1 db
2 progs
root@hpeos001[] #
```

This is not really an error; it's LVM just telling you that the volume group is still active. You will notice from the output above that the important part of this step is that the map file is created.

Something else to notice is that I haven't used the "`-s`" option to `vgexport`; this would write the concatenated "`CPU-ID+VG-ID`" into the map file. This seems like a good idea, but in my experience using this in a large configuration just causes you slight problems later on as we see.

6. You can now distribute the map file to all nodes in the cluster in preparation for using `vgimport` to import the relevant disks into the volume group.
7. The next problem is the fact that device files may be different on different nodes. The biggest problem is the Instance number of the interface the disk is connected to. Some administrators spend lots of time and effort to make the Instance numbers of all corresponding devices on a machine the same. That's fine by me if you want to take that route; see how we do it in Chapter 4, "Advanced Peripheral Configuration" (Rebuilding the **`ioinit`** File to Suit Your Needs). If you aren't going to spend all your time doing that, then you need to identify which disks are connected to which interfaces. You will need to work this out for all nodes in the cluster. See the example in Figure 25-2.

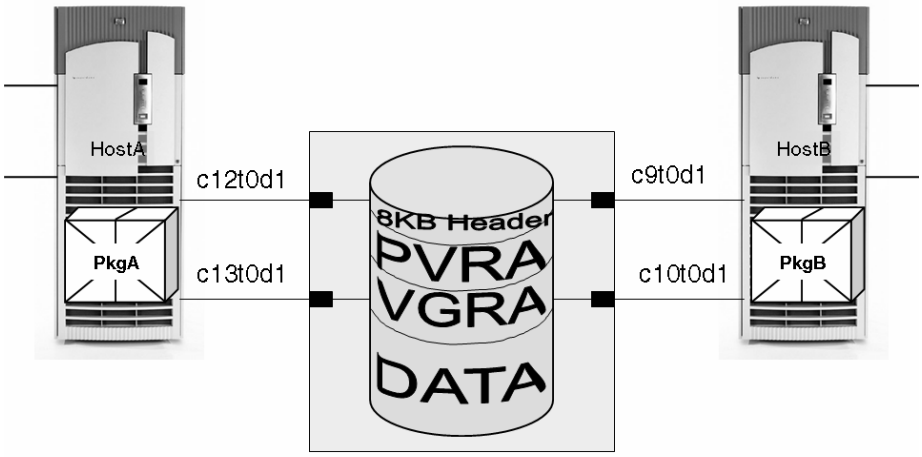

Figure 25–2 *Identifying shared disks.*

What we have is a single disk, dual-pathed from two hosts. In systems with lots of disks and multiple paths, e.g., through a SAN, you may have four paths per disk and possibly hundreds of disks. This example goes to show how you would identify which paths are "related" to a particular disk. Just looking at the device files will not show you much. Using a command like `diskinfo`

won't yield much information either. We need to go back to our understanding of the layout of an LVM disk. At the beginning of a disk, we have the 8KB header used to point to the boot area. We then have the PVRA. The first element of the PVRA is an LVM Record, which identifies this disk as an LVM disk. The LVM Record is 8 bytes in size. So take 8KB + 8 bytes = 8200 (2008 in hex) bytes. If we read from that location, we find four interesting numbers in the PVRA; the CPU-ID, the PV-ID, the CPU-ID (again) and the VG-ID. If we were to read this information from *any* of the device files, we should see *exactly* the same information. I would use the following command to do this:

```
# echo "0x2008?4X" | adb /dev/dsk/cXtYdZ
```

adb moves to the prescribed address and prints four integers; in my case, I display them in hex (the reason for using hex will become apparent). In fact, if you look at the output below, I have run just that command on my first node, hpeos001:

```
root@hpeos001[] # echo "0x2008?4X" | adb /dev/dsk/c12t0d1
2008: 77A22A2C 3C9DFCD9  77A22A2C  3C9DFCD6
root@hpeos001[] # echo "0x2008?4X" | adb /dev/dsk/c13t0d1
2008: 77A22A2C 3C9DFCD9  77A22A2C  3C9DFCD6
```

If we look at the output from the commands run from the other node, hpeos002, the output should be the same.

```
root@hpeos001[] # echo "0x2008?4X" | adb /dev/dsk/c9t0d1
2008: 77A22A2C 3C9DFCD9  77A22A2C  3C9DFCD6
root@hpeos001[] # echo "0x2008?4X" | adb /dev/dsk/c10t0d1
2008: 77A22A2C 3C9DFCD9  77A22A2C  3C9DFCD6
```

Theses are the two nodes as shown in Figure 25-2. As you can see, the relevant fields match up regardless of which device file we look at. In a large, complex environment, this can help you visualize which device files are "related" to which disks.

8. We can use vgimport to get the relevant disks into the relevant volume groups using the map file we distributed earlier.

```
# mkdir /dev/vg01
# mknod /dev/vg01/group c 64 0x010000
# vgimport /dev/vg01 /dev/dsk/c9t0d1 /dev/dsk/c10t0d1
# vgchange -a y /dev/vg01
# vgcfgbackup /dev/vg01
# vgchange -a n /dev/vg01
```

I want to say just a word regarding the "-s" option to vgexport/vgimport. I mentioned earlier that in my experience using this option would cause you slight problems. Here are the reasons why:

a. You are going to have to document the layout of your disks anyway, so why not do it now?

b. When we use the "`-s`" option with `vgimport`, `vgimport` must scan *every* disk on the system looking for matching `CPU-ID+VG-ID` values for corresponding disks. When you have *lots* of disks, this can take *many* seconds. In fact, on some systems I have seen it take many minutes. While it is inquiring of *all* those disks, you are interrupting other important data related IO.

c. **Conclusion:** Get to know your hardware; it makes sense in the long run.

9. It might be advisable to start to draw some form of diagram so that you know how your disks map to the device files. Documentation will certainly help later, as well as in any Disaster Recovery scenario.

• LAN cards

The important thing to remember here is the importance of having a bridged network between your "active" LAN cards and your "standby" LAN cards. Before embarking on creating a cluster, you must ensure that you can `linkloop` from and to each LAN card in your bridged network. If not, Serviceguard will not allow you to proceed. It's a relatively simple process as long as you are confident in how your network has been physically constructed. Let's look at a simple example shown in Figure 25-3.

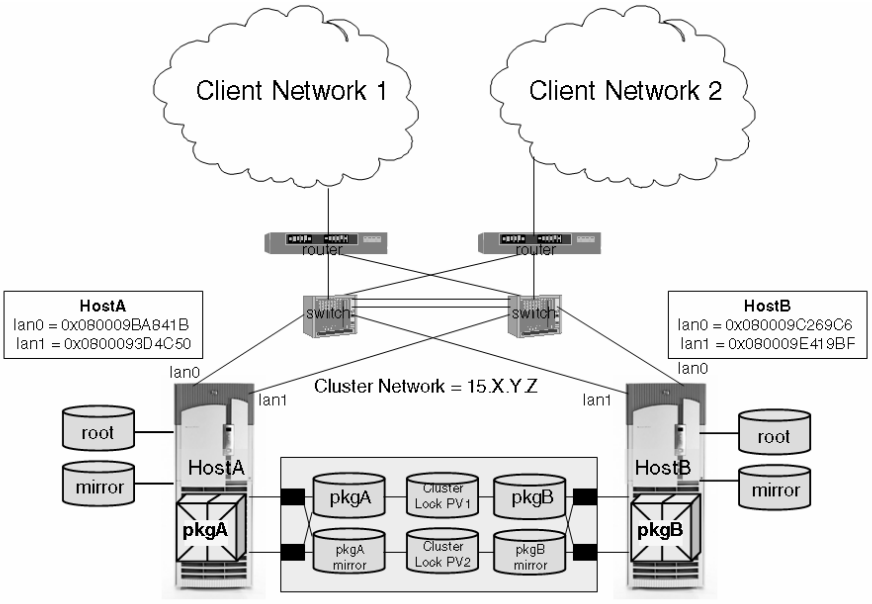

Figure 25–3 *A bridged network.*

We can see from Figure 25-3 that we have eliminated all the SPOFs for the Cluster Network. Now we need to check that our switch/hub/bridge has no filtering or blocking configured on the ports that we are using. If we are using two links between the switches, as shown, it may be a good idea to disconnect one link, perform the tests, swap the links over, and perform the tests again. It is not important which node you perform the test from because if one component is not working, it will show up. Some administrators perform the tests from both nodes "just to be sure." I don't mind that as a philosophy. I perform the tests on one node, and let you perform the tests on both of your nodes. Note that I am sending packets *out* of both interfaces and *to* both interfaces; so in our example, that means four `linkloop` tests. Below, I am performing the tests on two nodes configured similarly to the nodes in Figure 25-3.

```
root@hpeos001[] # lanscan
Hardware Station        Crd Hdw   Net-Interface  NM  MAC     HP-DLPI DLPI
Path     Address        In# State NamePPA        ID  Type    Support Mgr#
8/16/6   0x080009BA841B 0   UP    lan0 snap0     1   ETHER   Yes     119
8/20/5/2 0x0800093D4C50 1   UP    lan1 snap1     2   ETHER   Yes     119
root@hpeos001[] # linkloop -i 0 0x080009C269C6
Link connectivity to LAN station: 0x080009C269C6
 -- OK
root@hpeos001[] # linkloop -i 0 0x080009E419BF
Link connectivity to LAN station: 0x080009E419BF
 -- OK
root@hpeos001[] # linkloop -i 1 0x080009C269C6
Link connectivity to LAN station: 0x080009C269C6
 -- OK
root@hpeos001[] # linkloop -i 1 0x080009E419BF
Link connectivity to LAN station: 0x080009E419BF
 -- OK
```

I am going to assume that you have worked out all the SPOFs we talked about earlier and that you are happy with your current IT processes, system performance, user access, and security issues. I am also going to assume that all nodes in the cluster are listed in your host lookup database.

25.7 Setting Up a Serviceguard Package-less Cluster

We have finally arrived at creating the cluster. I think you will agree that all our previous discussions were worthwhile. Now that we understand the implications and requirements for setting up a Serviceguard cluster, we can proceed. By the end of this section, we have a running and tested *package-less* cluster. We use the cookbook we saw at the beginning of this chapter to set up the cluster and construct a number of tests to ensure that the cluster daemons are working as expected.

25.7.1 Understand the hardware and software implications of setting up a cluster

This topic was covered in the previous section. If you have *jumped* straight to this section, I think it would be beneficial for you to review the preceding six sections.

25.7.2 Set up NTP between all cluster members

The activities of all nodes in the cluster are now coordinated in an attempt to offer a unified computing resource. All nodes in the cluster should be using a common time source. It is a good idea if you set up Network Time Protocol (NTP). As a minimum, use one of the machines as the time source synchronizing with its local clock (a Local Clock Impersonator).

25.7.3 Ensure that any shared volume groups are not activated at boot time

We are now at the stage of coordinating activities between nodes. As we have mentioned, Serviceguard is a part of that process. An important part of Serviceguard's role it to coordinate the use of shared data. If we are using LVM volume groups, we need to ensure that our shared volume groups are not activated at system boot time. It is up to Serviceguard to activate a volume group on a node that needs access to the data. In order to disable volume group activation at boot time, we need to modify the startup script `/etc/lvmrc`. The first part of this process is as follows:

```
AUTO_VG_ACTIVATE=1
```

Changed to …

```
AUTO_VG_ACTIVATE=0
```

You then need to tell `/etc/lvmrc` which volume groups are to be activated at boot time. These will be volume groups that contain files and data that are unique for individual nodes, e.g., a node may have a volume group that contains application documentation and manuals. It does not include `vg00` because this is activated before `/etc/lvmrc` is referenced. Having your additional volume groups activated at boot time is accomplished in the function `custom_vg_activation()`. I have listed the entire function below with the line I edited underlined and bold.

```
custom_vg_activation()
{
        # e.g., /sbin/vgchange -a y -s
        #       parallel_vg_sync "/dev/vg00 /dev/vg01"
          parallel_vg_sync "/dev/vgmanuals"

        return 0
}
```

This needs to be performed on all nodes in the cluster. The list of volume groups that are to be activated will vary from machine to machine.

25.7.4 Install Serviceguard and any related Serviceguard patches

To get the most up-to-date patches for Serviceguard, browse on the Web to Hewlett-Packard's IT Resource Center at http://itrc.hp.com. Registration is free, and you'll get access to lots of useful information as well as access to the most up-to-date patches. The issue of patches is covered in Chapter 12, "HP-UX Patches." Patches should be installed after the base product is installed. Installing the base product is pretty straightforward; it's simply a case of a `swinstall`. Most recent versions of Serviceguard don't even need to reboot. Here are some commands to check whether it is installed:

```
# swlist -l fileset Serviceguard
# swlist -l fileset -s /cdrom -a is_reboot Serviceguard
```

If you are installing either the "Mission Critical" or "Enterprise" Operating Environment for HP-UX 11i, Serviceguard should be installed automatically.

It might be a good idea to make sure that you install two additional products at the same time:

- **EMS HA Monitors (version A.03.20.01 or later):** This usually entails at least six products on CD 2 of your 11i Core OS CD (alternatively on your Applications CD prior to 11i):
  - `EMS-Config`
  - `EMS-Core`
  - `EMS-DiskMonitor`
  - `EMS-KRMonitor`
  - `EMS-MIBMonitor`
  - `EMS-RdbmsMon`

  We use these for monitoring other critical resources.
- **Cluster Object Manager (version B.01.04 or later):** This usually entails a single product on the same CD as the EMS HA Monitors:
  - `Cluster-OM`

  This will be used later with the Serviceguard Manager product.

At this point, I downloaded and installed the Serviceguard patch `PHSS_28851` onto my systems. **NOTE: Check that you have the most up-to-date patch for your version of Serviceguard**:

```
root@hpeos001[oracle1] # swlist PHSS_28851
# Initializing...
# Contacting target "hpeos001"...
#
# Target:  hpeos001:/
```

```
#

# PHSS_28851                    1.0            Serviceguard and SG-OPS Edition
A.11.14
  PHSS_28851.ATS-MAN          1.0          Service Guard Advanced Tape Services

  PHSS_28851.ATS-RUN          1.0          Service Guard Advanced Tape Services

  PHSS_28851.CM-CORE           1.0            CM-CORE Serviceguard OPS Edition
SD fileset
  PHSS_28851.CM-CORE-MAN       1.0            CM-CORE-MAN Serviceguard OPS Edi-
tion SD fileset
  PHSS_28851.CM-PKG            1.0           CM-PKG Serviceguard OPS Edition SD
fileset
  PHSS_28851.CM-PKG-MAN        1.0           CM-PKG-MAN Serviceguard OPS Edition
SD fileset
root@hpeos001[oracle1] #
```

25.7.5 Installing a Quorum Server (optional in a basic cluster)

This is optional, but it's becoming more common. First, you need to choose a system or systems where you will run the Quorum Server software. I say "systems" because someone quite kindly pointed out that a single Quorum Server could be seen as an SPOF. To alleviate this, I would run my Quorum Server in a different and separate Serviceguard cluster and eliminate the SPOF by configuring a package that managed the Quorum Serve application. If the primary Quorum Server node fails, it will fail over to an adoptive node. The fact that we can associate an IP address with an application means that our original cluster maintains contact with the Quorum Server via its package IP address. As an example, if you had "Finance" and "Sales" clusters, each could run a Quorum Server Package for the other cluster! Now back to the installation:

1. Choose a node or nodes that are *not* part of this cluster. The nodes can be running either HP-UX or Linux.
2. **Install the Quorum Server software (B8467BA version A.02.00):**
   a. Either from the HP Serviceguard Distributed Components CD, or
   b. Download the product for free from http://software.hp.com, under "High Availability." The last time I looked, it was titled "Serviceguard Quorum Server." Use the appropriate tool, i.e., `swinstall` or `rpm` to install the product.
3. The installation doesn't put an entry into `/etc/inittab` (some installations will configure the Quorum Server software as a Serviceguard package that can subsequently move to an adoptive node should the original Quorum Server fail). You are going to have to do that yourself to ensure that `/usr/lbin/qs` (`/usr/local/qs/bin/qs` for Linux) gets started at boot-up time and gets restarted (the `respawn` action in `/etc/inittab`) if necessary. The entry should look something like this:

```
qs:345:respawn:/usr/lbin/qs >> /var/adm/qs/qs.log 2>&1
```

It might be a good idea to ensure that the `/var/adm/qs` directory has been created as well. Don't run `init q` yet because the `qs` daemon will refuse to start if you don't have an authorization file in place.

4. Set up an authorization file of all nodes requiring Quorum Services. It is simply a list of hostnames and/or IP addresses (why not put both!), one per line; ensure that all nodes in the cluster are entered in the authorization file. The authorization file is called `/etc/cmcluster/qs_authfile` (or `/usr/local/qs/conf/qs_authfile` for Linux).
5. Now we can start the `qs` daemon by running `init q`.
6. Check that the `qs` daemon is running by monitoring the file `/var/adm/qs/qs.log`. Here's the output from a machine that successfully started the `qs` daemon.

```
Apr 05 16:51:32:0:Starting quorum server
Apr 05 16:51:32:0:Total allocated: 440788 bytes, used: 20896 bytes, unused
  419880 bytes
Apr 05 16:51:32:0:Server is up and waiting for connections at port 1238
```

   a. If you need to update the authorization file, e.g., you add a new node to the cluster, ensure that you get the `qs` daemon to reread the authorization file. To do this, you simply run the `qs` command with the `–update` command line argument.
      For HP-UX: `# /usr/lbin/qs -update`
      For Linux: `# /usr/local/qs/bin/qs –update`

25.7.6 Enable remote access to all nodes in the cluster

Early versions of Serviceguard required the use of the file `$HOME/.rhosts` for the root user. This obviously had implications for network security. To get around this, Serviceguard offers an alternative. You can use a file called `/etc/cmcluster/cmclnodelist`. The format of this file is the same as `$HOME/.rhosts`:

```
<hostname> <username>
```

You should list all hostnames in the cluster, and the username will likely be `root`. If you want other users to be able to monitor the cluster, list their hostname/username as well. Oh, another thing. It might be a good idea to list the IP address/username as well. Just in case something happens to your host lookup database, e.g., DNS, you don't want that causing you unnecessary problems when managing your cluster. Make sure *every* node has the `/etc/cmcluster/cmclnodelist` file in place and a *complete* list of hostname/username and IP address/username for all machines in the cluster.

25.7.7 Create a default ASCII cluster configuration file

This is easier than you think. You simply need to ensure that all nodes are running and contactable, and that the `cmclnodelist` file is in place. Let's log in to one of the nodes in the cluster. It doesn't matter which one. This is where we will initially perform all of our configuration steps. Serviceguard supplies a command (`cmquerycl`) that will probe all the nodes we tell it to. It will work out for itself which LAN cards are active, which LAN cards "*could*" be standby LAN cards, and which volume groups are shared; it will list the first physical volume as a candidate as a Cluster Lock PV. If we have the Quorum Server up and running, it can even fill in those details as well. Here are a couple of examples:

```
# cd /etc/cmcluster
# cmquerycl -v -C cluster.ascii -n node1 -n node2
```

The `-v` (verbose) is optional, but it does give you some idea what `cmquerycl` is doing. The `-C` just specifies the filename to store the resulting configuration file. The `-n` specifies the nodes to be included in the cluster. Not too challenging, is it? Alternately, you can specify a Quorum Server (`-q qshost1` in our example) and give your cluster a name (`-c McBond` in our example) up front:

```
# cd /etc/cmluster
# cmquerycl -v -C cluster.ascii -c McBond -q qshost1 -n node1 -n node2
```

This takes a few seconds to complete. Errors will have to be resolved before moving on.

25.7.8 Update the ASCII cluster configuration file

I ran the first of the example `cmquerycl` commands above. I have listed below the content of the `cluster.ascii` configuration file.

```
# **********************************************************************
# ********* HIGH AVAILABILITY CLUSTER CONFIGURATION FILE ***************
# ***** For complete details about cluster parameters and how to    ****
# ***** set them, consult the Serviceguard manual. ****
# **********************************************************************

# Enter a name for this cluster. This name will be used to identify the
# cluster when viewing or manipulating it.

CLUSTER_NAME            cluster1


# Cluster Lock Parameters
#
# The cluster lock is used as a tiebreaker for situations
# in which a running cluster fails, and then two equal-sized
# sub-clusters are both trying to form a new cluster. The
# cluster lock may be configured using either a lock disk
```

```
# or a quorum server.
#
# You can use either the quorum server or the lock disk as
# a cluster lock but not both in the same cluster.
#
# Consider the following when configuring a cluster.
# For a two-node cluster, you must use a cluster lock. For
# a cluster of three or four nodes, a cluster lock is strongly
# recommended. For a cluster of more than four nodes, a
# cluster lock is recommended. If you decide to configure
# a lock for a cluster of more than four nodes, it must be
# a quorum server.

# Lock Disk Parameters. Use the FIRST_CLUSTER_LOCK_VG and
# FIRST_CLUSTER_LOCK_PV parameters to define a lock disk.
# The FIRST_CLUSTER_LOCK_VG is the LVM volume group that
# holds the cluster lock. This volume group should not be
# used by any other cluster as a cluster lock device.

# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL,
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.
# The QS_HOST is the host name or IP address of the system
# that is running the quorum server process. The
# QS_POLLING_INTERVAL (microseconds) is the interval at which
# Serviceguard checks to make sure the quorum server is running.
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase
# the time interval after which the quorum server is marked DOWN.
#
# The default quorum server timeout is calculated from the
# Serviceguard cluster parameters, including NODE_TIMEOUT and
# HEARTBEAT_INTERVAL. If you are experiencing quorum server
# timeouts, you can adjust these parameters, or you can include
# the QS_TIMEOUT_EXTENSION parameter.
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000
# QS_TIMEOUT_EXTENSION 2000000

FIRST_CLUSTER_LOCK_VG           /dev/vg01


# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.

NODE_NAME               hpeos001
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP                192.168.0.201
  NETWORK_INTERFACE     lan1
```

```
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c1t0d0
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

NODE_NAME               hpeos002
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP                192.168.0.202
  NETWORK_INTERFACE     lan1
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c0t0d0
# List of serial device file names
# For example:
# SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.


# Cluster Timing Parameters (microseconds).
# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
# This default setting yields the fastest cluster reformations.
# However, the use of the default value increases the potential
# for spurious reformations due to momentary system hangs or
# network load spikes.
# For a significant portion of installations, a setting of
# 5000000 to 8000000 (5 to 8 seconds) is more appropriate.
# The maximum value recommended for NODE_TIMEOUT is 30000000
# (30 seconds).

HEARTBEAT_INTERVAL              1000000
NODE_TIMEOUT            2000000

# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT      600000000
NETWORK_POLLING_INTERVAL        2000000

# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the
# cluster.
# You can not add packages beyond this limit.
# This parameter is required.
MAX_CONFIGURED_PACKAGES         0

# List of cluster aware LVM Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# Neither CVM or VxVM Disk Groups should be used here.
# For example:
# VOLUME_GROUP          /dev/vgdatabase
# VOLUME_GROUP          /dev/vg02

VOLUME_GROUP            /dev/vg01
```

Thankfully, `cmquerycl` takes most of the heartache out of constructing this file. As you can see, there's quite a lot to digest. A formal definition of all these parameters can be found in the Serviceguard documentation. Here, I attempt to put the formal definition into more meaningful English. The file can be broken down into a number of sections:

1. **Cluster name**
   - — Use a name that means something to you. You might have a naming convention for things such as hostnames, so it might be appropriate to have a similar naming convention if you are going to run multiple clusters.
2. **Cluster lock strategy,** i.e., LVM disk and/or Quorum Server
   - — The name of the Quorum Server.
   - — Timing parameters for the Quorum Server. Every two minutes, the Quorum Server is polled to make sure it's still alive. If you have a busy network, you can use the "*extension*" parameter to extend the polling interval, or simply increase the polling interval itself.
   - — We also specify the name of the LVM volume group that holds the cluster lock disk. The parameter is called `FIRST_CLUSTER_LOCK_VG`. If your disks are powered from the same power supply as the nodes themselves, you may consider using a `SECOND_CLUSTER_LOCK_VG`. We could then specify a `SECOND_CLUSTER_LOCK_PV` for each node.
3. **Individual node specifications**
   - — We specify the node name as well as all LAN interfaces for that node.
   - — Remember, a LAN interface can be either `HEARTBEAT_IP` or `STATIONARY_IP`. `STATIONARY_IP` means that we don't send heartbeat packets over that interface. The default assumed by `cmquerycl` is to specify `HEARTBEAT_IP` for all active interfaces.
   - — We list the device file for the `FIRST` and possibly `SECOND_CLUSTER_LOCK_PV`.
   - — We specify that the device file we are using is a `SERIAL_HEARTBEAT`.
4. **Cluster timing parameters**
   - — `HEARTBEAT_INTERVAL` specifies how often a heartbeat packet is transmitted. The default of 1 second seems reasonable.
   - — `NODE_TIMEOUT` defaults to 2 seconds. This is the time at which a node is determined to have failed. If you leave it at 2 seconds, you will get a warning that it might be a good idea to increase this value. The dilemma is the fact that we would like quick cluster reformations when a node does actually fail, hence the 2 second default. Unfortunately, if we have a busy network, then heartbeat packets might not get through in time and we would experience a cluster reformation. The reality is that after the reformation, all the nodes will still be online so nothing drastic will actually happen. The problem is that if you are monitoring `syslog.log` and you see a cluster reformation, it might ring some *alarm bells*. It's up to you; my

thinking is that if you have a dedicated heartbeat LAN, then 2 seconds should be okay as the maximum traffic on that LAN would be 16 (maximum nodes in a cluster) nodes sending a heartbeat packet once every second; that's not much traffic.

5. **Configuration/reconfiguration timing parameters**
   — The first time we start the cluster, we must have all nodes online, i.e., 100 percent node attendance. The `AUTOSTART_TIMEOUT` (default = 10 minutes) is how long we wait for other nodes to become available. Otherwise, we don't start the cluster.
   — `NETWORK_POLLING_INTERVAL` (default 2 seconds) is how often `cmcld` checks that our standby LAN cards are still working as specified. If we lose an active LAN card, `cmcld` will immediately move the IP address to the standby LAN card, so it's a good idea to check that the bridged network is operational every few seconds.
6. **Package configuration parameters**
   — I want to point out only one thing here: `MAX_CONFIGURED_PACKAGES` (default = 0). That's fairly easy to understand. If we are to configure more packages than this parameter allows, we would need to shut down the entire cluster, so choose a reasonable value up front.
7. **LVM volume groups**
   — This is a list of LVM volume groups that will be marked as "*cluster aware*" (`vgchange -c y`) as soon as the `cmcld` daemon is run. If you try this command without `cmcld` running, you get an error message. You don't need to list your volume groups here, but make sure when you start the `cmcld` daemon that you run `vgchange -c y` against all your shared volume groups. Being "*cluster aware*" sets a flag in the VGRA that allows the volume group to be activated in "*exclusive*" mode (`vgchange -a e`). Here are the changes I made to this file:
   a. Change the default cluster name:

```
CLUSTER_NAME            McBond
```

   b. Set up a serial heartbeat if applicable. This is detailed in the individual node specifications:

```
SERIAL_DEVICE_FILE    /dev/tty0p0
SERIAL_DEVICE_FILE    /dev/tty1p0
```

   c. Allow me to create packages in the future:

```
MAX_CONFIGURED_PACKAGES         10
```

25.7.9 Check the updated ASCII cluster configuration file

We need to ensure that all our changes are syntactically correct and can actually be applied to the nodes in question. We have a simple command to do this: `cmcheckconf`. Below, you see me run this command and the output I received:

```
root@hpeos001[cmcluster] # cmcheckconf -v -C cluster.ascii

Checking cluster file: cluster.ascii
Note : a NODE_TIMEOUT value of 2000000 was found in line 104. For a significant
portion of installations, a higher setting is more appropriate.
Refer to the comments in the cluster configuration ascii file or Serviceguard
manual for more information on this parameter.
Checking nodes ... Done
Checking existing configuration ... Done
Warning: Can not find configuration for cluster McBond
Gathering configuration information ... Done
Gathering configuration information ..............  Done
Checking for inconsistencies .. Done
Maximum configured packages parameter is 10.
Configuring 0 package(s).
10 package(s) can be added to this cluster.
Creating the cluster configuration for cluster McBond.
Adding node hpeos001 to cluster McBond.
Adding node hpeos002 to cluster McBond.

Verification completed with no errors found.
Use the cmapplyconf command to apply the configuration.
root@hpeos001[cmcluster] #
```

Any errors need to be corrected before we move on.

25.7.10 Compile and distribute binary cluster configuration file

We can now compile the binary cluster configuration file `/etc/cmcluster/cmclconfig` from our ASCII template file. As you can see in the output from the `cmcheckconf` command, we use `cmapplyconf`. This will also distribute `cmclconfig` to all nodes in the cluster. One thing to be aware of is in connection with cluster lock disks. If you are using a cluster lock disk, it is a good idea to activate the volume group on the node from which you are running `cmapplyconf`. If you don't, `cmapplyconf` will attempt to activate it in exclusive mode in order to initialize the cluster lock information. If this fails, `cmapplyconf` will display an error, so having the volume group active in the first place avoids this error. Again, here is the output I received:

```
root@hpeos001[cmcluster] # cmapplyconf -v -C cluster.ascii

Checking cluster file: cluster.ascii
Note : a NODE_TIMEOUT value of 2000000 was found in line 104. For a significant
portion of installations, a higher setting is more appropriate.
```

```
Refer to the comments in the cluster configuration ascii file or Serviceguard
manual for more information on this parameter.
Checking nodes ... Done
Checking existing configuration ... Done
Warning: Can not find configuration for cluster McBond
Gathering configuration information ... Done
Gathering configuration information ............. Done
Checking for inconsistencies .. Done
Maximum configured packages parameter is 10.
Configuring 0 package(s).
10 package(s) can be added to this cluster.
Creating the cluster configuration for cluster McBond.
Adding node hpeos001 to cluster McBond.
Adding node hpeos002 to cluster McBond.
Completed the cluster creation.
root@hpeos001[cmcluster] #
```

Before we start the cluster, we need to ensure that we have a consistent backup of the LVM structures.

25.7.11 Back up LVM structures of any cluster lock volume groups

The LVM structures in the Volume Group Reserved Area (VGRA) deal with the state and location on disk of the cluster lock. The actual cluster lock and who currently owns it is stored in the Bad Block Relocation Area (BBRA) of the disk. Although we won't back up the actual cluster lock itself, we should back up the LVM structures that relate to it in the VGRA. This is why we use `vgcfgbackup` at this time. Should we need to, i.e., if a cluster lock disk fails, we can recover these fields with `vgcfgrestore`.

In our case, it is simply a case of running the following command:

```
root@hpeos001[] # vgcfgbackup /dev/vg01
Volume Group configuration for /dev/vg01 has been saved in /etc/lvmconf/
vg01.conf
root@hpeos001[] #
```

We should consider storing the `vgcfgbackup` file (`/etc/lvmconf/vg01.conf` in this case) on all nodes in the cluster. Because Serviceguard is responsible for activating and deactivating volume groups, we should deactivate this volume group with the following:

```
root@hpeos001[] # vgchange -a n /dev/vg01
vgchange: Volume group "/dev/vg01" has been successfully changed.
root@hpeos001[] #
```

We are now ready to start the cluster.

25.7.12 Start cluster services

Ensure that all nodes are online before attempting to start the cluster for the first time. We need 100 percent node attendance. Here is the output from my cluster:

```
root@hpeos001[cmcluster] # cmruncl -v
Successfully started $SGLBIN/cmcld on hpeos001.
Successfully started $SGLBIN/cmcld on hpeos002.
cmruncl  : Waiting for cluster to form.....
cmruncl  : Cluster successfully formed.
cmruncl  : Check the syslog files on all nodes in the cluster
cmruncl  : to verify that no warnings occurred during startup.
root@hpeos001[cmcluster] #
```

The command `cmruncl` is how we start cluster services when the cluster is down. When we start the cluster, it would be helpful to have all nodes online. This is not always possible, i.e., one node is down due to urgent maintenance. In this situation, we could use the option `-n <nodename>` to `cmruncl`, listing all nodes that are currently available. Here's what happens if I use this command to start cluster services on one node (the other node `hpeos001` is currently down):

```
root@hpeos002[] # cmruncl -v -n hpeos002

WARNING:
Performing this task overrides the data integrity protection normally provided by
Serviceguard. You must be certain that no package applications or resources are
running on the other nodes in the cluster:
       hpeos001

To ensure this, these nodes should be rebooted (i.e., /usr/sbin/shutdown -r)
before proceeding.

Are you sure you want to continue (y/[n])?
```

The reason for the warning is that if the down node(s) restarted but had network problems so that they couldn't contact the nodes currently in the cluster, they could potentially form a cluster of their own. This could lead to two sets of nodes trying to start the same applications. This is not a good idea. Once the down system is rebooted, we can have that node join the cluster with the command `cmrunnode`.

We should start to get into the habit of running `cmviewcl -v`. As you can gather, I like my `-v` option. In a cluster, you probably want to know that *everything* is working as expected. You can use a `-n nodename` just to view specific nodes. Here is the output from my `cmviewcl` command:

```
root@hpeos001[cmcluster] # cmviewcl -v

CLUSTER      STATUS
McBond       up

  NODE         STATUS       STATE
  hpeos001     up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
```

```
   STANDBY      up           8/20/5/2     lan1

NODE          STATUS       STATE
hpeos002      up           running

   Network_Parameters:
   INTERFACE    STATUS       PATH         NAME
   PRIMARY      up           2/0/2        lan0
   STANDBY      up           4/0/1        lan1
```

Let's look at `/var/adm/syslog/syslog.log`. This will detail the starting of the cluster. It takes time to get used to the output from different cluster operations. That's why we are going to test cluster functionality at the end of this section. Part of that will be to check `syslog.log` and find out what is happening. Here's the healthy output I received in my `syslog.log` when I started the cluster:

```
root@hpeos001[cmcluster] # more /var/adm/syslog/syslog.log
…
Aug  2 15:48:57 hpeos001 CM-CMD[2733]: cmruncl -v
Aug  2 15:48:58 hpeos001 inetd[2734]: hacl-cfg/udp: Connection from localhost
(127.0.0.1) at Fri Aug  2 15:48:58 2002
Aug  2 15:48:58 hpeos001 inetd[2735]: hacl-cfg/tcp: Connection from localhost
(127.0.0.1) at Fri Aug  2 15:48:58 2002
Aug  2 15:48:58 hpeos001 inetd[2736]: hacl-cfg/tcp: Connection from localhost
(127.0.0.1) at Fri Aug  2 15:48:58 2002
Aug  2 15:48:58 hpeos001 cmclconfd[2736]: Executing "/usr/lbin/cmcld" for node
hpeos001
Aug  2 15:48:58 hpeos001 inetd[2738]: hacl-cfg/tcp: Connection from localhost
(127.0.0.1) at Fri Aug  2 15:48:58 2002
Aug  2 15:48:58 hpeos001 cmcld: Daemon Initialization - Maximum number of pack-
ages supported for this incarnation is 10.
Aug  2 15:48:58 hpeos001 cmcld: Global Cluster Information:
Aug  2 15:48:58 hpeos001 cmcld: Heartbeat Interval is 1 seconds.
Aug  2 15:48:58 hpeos001 cmcld: Node Timeout is 2 seconds.
Aug  2 15:48:58 hpeos001 cmcld: Network Polling Interval is 2 seconds.
Aug  2 15:48:58 hpeos001 cmcld: Auto Start Timeout is 600 seconds.
Aug  2 15:48:58 hpeos001 cmcld: Information Specific to node hpeos001:
Aug  2 15:48:58 hpeos001 cmcld: Cluster lock disk: /dev/dsk/c1t0d0.
Aug  2 15:48:58 hpeos001 cmcld: lan0  0x080009ba841b  192.168.0.201  bridged
net:1
Aug  2 15:48:58 hpeos001 inetd[2739]: hacl-cfg/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 15:48:58 2002
Aug  2 15:48:58 hpeos001 cmcld: lan1  0x0800093d4c50    standby    bridged net:1
Aug  2 15:48:58 hpeos001 cmcld: Heartbeat Subnet: 192.168.0.0
Aug  2 15:48:58 hpeos001 cmcld: The maximum # of concurrent local connections to
the daemon that will be supported is 38.
Aug  2 15:48:59 hpeos001 cmcld: Total allocated: 2097832 bytes, used: 3726072
bytes, unused 2017224 bytes
Aug  2 15:48:59 hpeos001 cmcld: Starting cluster management protocols.
Aug  2 15:48:59 hpeos001 cmcld: Attempting to form a new cluster
Aug  2 15:49:00 hpeos001 cmtaped[2743]: cmtaped: There are no ATS devices on this
cluster.
```

```
Aug  2 15:49:01 hpeos001 cmcld: New node hpeos002 is joining the cluster
Aug  2 15:49:01 hpeos001 cmcld: Clearing Cluster Lock
Aug  2 15:49:01 hpeos001 inetd[2749]: hacl-cfg/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 15:49:01 2002
Aug  2 15:49:03 hpeos001 cmcld: Turning on safety time protection
Aug  2 15:49:03 hpeos001 cmcld: 2 nodes have formed a new cluster, sequence #1
Aug  2 15:49:03 hpeos001 cmcld: The new active cluster membership is:
hpeos001(id=1), hpeos002(id=2)
Aug  2 15:49:03 hpeos001 cmlvmd: Clvmd initialized successfully.
Aug  2 15:49:03 hpeos001 inetd[2750]: hacl-cfg/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 15:49:03 2002
Aug  2 15:49:03 hpeos001 inetd[2751]: hacl-cfg/tcp: Connection from hpeos002
(192.168.0.202) at Fri Aug  2 15:49:03 2002
Aug  2 15:49:04 hpeos001 inetd[2752]: hacl-cfg/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 15:49:04 2002
Aug  2 15:49:16 hpeos001 inetd[2753]: registrar/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 15:49:16 2002
```

As you can see, there is quite a lot going on. Basically, we start the `cmcld` daemon. In initializing, `cmcld` outputs our cluster timing parameters. It then identifies which LAN cards are active and which are Standby cards. We then work out whether there are any shared tape devices. We then see the other node (`hpeos002`) joining the cluster, giving us two members. Finally, the cluster LVM daemon is started. The entries you see for `hacl-cfg` come from the cluster configuration daemon (`cmclconfd`) that gathers information about LAN cards and volume groups. It also distributes the cluster binary file. During the startup of the cluster, all nodes are communicating with each other to ensure that the cluster is formed correctly and also to elect a **cluster coordinator**. If `cmcld` needs to gather information, it will do so by making a request to a `cmclconfd` process—actually to a network socket being managed by `inetd`. `inetd` is listening for requests on port 5302, and it is `inetd` that will actually spawn the `cmclconfd` daemon. One problem I have seen in the past is the entries in `/etc/inetd.conf` were missing. This causes weird results; I once saw "`Error: Unable to establish communication to node <nodename>`" when executing a `cmquerycl`. We checked everything from cables to `linkloop` commands and tried resetting LAN cards with `lanadmin`. The only reason I managed to fix it was that my suspicions were aroused by the lack of entries in `syslog.log` for `cmclconfd`. In the end, the customer involved admitted that he had recently uninstalled and reinstalled Serviceguard *a few times*. Don't as me why, he just did. The key was getting familiar with the expected output in `syslog.log` and trying to troubleshoot from first principles. We should see similar output on all nodes in the cluster.

Here is a brief overview of the election protocol every time a cluster reforms:

1. Start a reconfiguration timer.
2. Search for the existing cluster coordinator.
   — Send an FC_broadcast (FindCoordinator) message and wait for a reply.
3. If the Cluster Coordinator replies, send them your "vote."
4. If no Cluster Coordinator replies, attempt to become the Cluster Coordinator.
   — Reply to other nodes and accept "votes."

5. After "election timeout," count the "votes."
   — <50%: Retry until "reconfiguration timeout" expires. If still <50%, halt the node.
   — =50%: Attempt to grab the cluster lock and form the cluster. If this fails, halt the node.
   — >50%: Form the cluster.
6. Wait for "quiescence" timeout, an elapsed time to allow other nodes to halt.
7. New Cluster Coordinator informs the cluster members of the status and membership of the cluster.
8. Start heartbeat packets to all cluster members.
9. Clear the cluster lock.
   Note: The current Cluster Coordinator does not perform steps (b) and (c).

If you are interested in finding the cluster coordinator, you need to increase the Serviceguard logging level. This is achieved by using the contributed command `cmsetlog`. Use of this command by customers is normally only under the guidance of HP Support personnel. HP does not offer official support for this command, so be very careful if you are going to use it. The command and its options are discussed on various HP Education Services courses covering Serviceguard. If you are unfamiliar with the command, it is strongly suggested you *do not* use it. We will need to increase the logging level to 4 (`/usr/contrib/bin/cmsetlog 4`) if we want Serviceguard to report which node is a cluster coordinator. The node that becomes the cluster coordinator will write a message into its `/var/adm/syslog/syslog.log` of the form "`Aug 2 17:22:57 hpeos001 cmcld: This node is now cluster coordinator`".

One last thing. We have the option of starting cluster services every time a node starts up. This is accomplished by editing the startup script `/etc/rc.config.d/cmcluster`. The default is to *not* start cluster services at boot time (`AUTOSTART_CMCLD=0`), and I agree with this for this reason: Once started, why would a node need rebooting? If it does reboot, I would want to know why. Let's look at an example of when a node crashes due to a hardware problem. It reboots, and if we set `AUTOSTART_CMCLD=1`, it rejoins the cluster. This will cause a cluster reformation. If the hardware problem is intermittent, the fault may not occur for some time. Alternately, it could happen almost immediately. With `AUTOSTART_CMCLD=1`, the node would be joining, leaving, and rejoining the cluster every few minutes. A cluster reformation in itself is not too much to ask, but it is something we want to avoid if it at all possible. Having spurious reformations can confuse everyone involved and may actually "hide" real problems when they do occur. With `AUTOSTART_CMCLD=0`, a node will stay out of the cluster, allowing you to investigate why it rebooted before having the node rejoin the cluster when the problem has been rectified.

25.7.13 Test cluster functionality

There are a number of tests we will perform. Some of them are quite straightforward and test the basic functionality of the cluster; we use Serviceguard commands to accomplish

these tests. I will call these **Standard Tests**. Other tests are designed to uncover whether Serviceguard can provide the high availability features it claims it can. For these tests, we use "unorthodox" methods to test Serviceguard. I call these **Stress Tests**. We need to be sure that Serviceguard will react promptly and correctly in the event of an unexpected incident, e.g., if a LAN card fails. Let's start with the Standard Tests:

1. **Standard Tests**:
   a. **Cluster can start and stop successfully.**

      You should be able to run the following to start the cluster:

      ```
      # cmruncl -v
      ```

      You should be able to run the following to halt the cluster:

      ```
      # cmhaltcl -v
      ```

      You can run these commands from any node in the cluster. Check the output in `/var/adm/syslog/syslog.log` to ensure that everything is working as expected. This is basic functionality. Do not proceed until you are satisfied that the cluster can be started and stopped from every node.

   b. **Individual nodes can leave the cluster.**

      When we are performing critical maintenance on an individual node, we want to stop cluster services only on that node. Some administrators feel that if the node is going to be "out of commission" for a considerable time, then we should take it out of the cluster altogether. I can see some logic in that. My only concern is that we will have to recompile the cluster binary configuration file to remove and then add the node into the cluster. What would happen if another node were not running during this recompilation? We could be in a position where we want to re-add the original node, but we are having to wait until the second node comes back online to ensure that every node has the most up-to-date cluster binary file. For this reason alone, I would leave the node as being a member of the cluster, but just stop cluster services. Even if we reboot the node, it will not start cluster services as `AUTOSTART_CMCLD=0`. To stop cluster service, we would run the following command:

      ```
      # cmhaltnode -v
      ```

      Ensure that cluster service have stopped by checking `/var/adm/syslog/syslog.log` and the output from `cmviewcl -v`. We could run `cmhaltnode` from any node in the cluster. If we want to halt cluster services for a node other than our own, we can run this:

      ```
      # cmhaltnode -v othernode
      ```

Obviously, `othernode` is the hostname on which we are starting up cluster services. Again, check `/var/adm/syslog/syslog.log` and the output from `cmviewcl -v` to ensure that everything is functioning as expected.

c. **Individual nodes can join the cluster.**
In this instance, we want a node to rejoin a running cluster. Maybe we have concluded our critical maintenance, or the machine crashed and we have finished our investigations and repairs. We want to start cluster services only on this node. To accomplish this, we run the following:

```
# cmrunnode -v
```

Ensure that cluster service has stopped by checking `/var/adm/syslog/syslog.log` and the output from `cmviewcl -v`. Like `cmhaltnode`, we could run `cmrunnode` from any node in cluster. If we want to start cluster services for a node other than our own, we can run this:

```
# cmrunnode -v othernode
```

Obviously, `othernode` is the hostname on which we are shutting down cluster services. Again, check `/var/adm/syslog/syslog.log` and the output from `cmviewcl -v` to ensure that everything is functioning as expected.

2. **Stress Tests:**
These test are a little "unorthodox" only insofar as we are trying to think of situations that may happen in a production environment and which could threaten access to our applications. We want to test these situations in a controlled way to ensure that Serviceguard is behaving as expected.

a. **Remove an active LAN card.**
There should be no perceived problems when we perform this test. Serviceguard should automatically relocate the IP address associated with our Primary LAN to the Standby LAN card. Serviceguard will also send out an ARP broadcast to all machines currently communicating via that IP address to flush their ARP cache and, hence, disassociate the IP address with a MAC address. All clients will then need to send an ARP request to reestablish the IP-MAC mapping. In doing so, they will now find that the MAC address of the Standby LAN card is associated with the relevant IP address. This is the output I found in `/var/adm/syslog/syslog.log` after I pulled the cable from my active LAN card and then put it back in:

```
Aug  2 19:39:19 hpeos001 cmcld: lan0 failed
Aug  2 19:39:19 hpeos001 cmcld: Subnet 192.168.0.0 switched from lan0
  to lan1
Aug  2 19:39:19 hpeos001 cmcld: lan0 switched to lan1
```

As we can see, Serviceguard reacted instantaneously to relocate the IP address to the standby LAN card. One word of warning: If you keep your

NODE_TIMEOUT value low, i.e., 2 seconds, it may be that you see a cluster reformation in `syslog.log` at the same time as Serviceguard relocates the IP address. This is due to timing issues with sending and receiving heartbeat packets. Because Serviceguard can relocate the IP address almost instantaneously, we see the cluster reform at the same time as the IP address is relocated. Here's what we see with `cmviewcl -v`:

```
root@hpeos001[cmcluster] # cmviewcl -v

CLUSTER      STATUS
McBond       up

  NODE         STATUS       STATE
  hpeos001     up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      down         8/16/6       lan0
    STANDBY      up           8/20/5/2     lan1

  NODE         STATUS       STATE
  hpeos002     up           running

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           2/0/2        lan0
    STANDBY      up           4/0/1        lan1
root@hpeos001[cmcluster] #
```

Notice that the `PRIMARY` LAN card for `hpeos001` is now "*down*." On reconnecting the LAN card, Serviceguard relocates the IP address back to the Primary LAN card, as we can see from `syslog.log`.

```
Aug  2 19:45:22 hpeos001 cmcld: lan0 recovered
Aug  2 19:45:22 hpeos001 cmcld: Subnet 192.168.0.0 switched from lan1
  to lan0
Aug  2 19:45:22 hpeos001 cmcld: lan1 switched to lan0
```

If you were seeing lots of these "local LAN failover" errors, then I would consider logging a Hardware Support Call with your local Hewlett-Packard Response Center to have a Hewlett-Packard Hardware Engineer check whether your LAN card is malfunctioning. It could also be a faulty cable or a faulty hub/switch.

b. **A situation where `cmcld` is starved for resources.**
This is a particularly critical situation. As we now know, `cmcld` is a critical part of the suite of Serviceguard daemons. It is considered to be so important that it runs at an HP-UX Real-Time Priority of 20. This means that when it wants to run, there's a high probability that it will be the most important process on the

system. There are few processes with a higher priority. However, I have come across many installations where Real-Time priorities have been used to improve the responsiveness of critical application processes. In one such situation—a four-processor machine—the administrators had four database instances running in a Serviceguard cluster. The main database daemons were running at priority = 20 in an attempt to maximize the amount of CPU time the main database processes received. The administrators felt that it was highly unlikely that at any one time all the database processes would be executing requests to such an intensity that `cmcld` would not get execution time on any processor. As we know from Murphy's Law, such a situation did arise. The database processes spawned a significant number of child processes. Along with `cmcld`, this constituted enough of a contention that `cmcld` did not get any execution time in the `NODE_TIMEOUT` interval. The cluster coordinator made a decision that the node had failed and instigated a cluster reformation. On reforming the cluster (a two-node cluster), the original node had, by that time, "*resolved*" its starvation problem and won the resulting election and, hence, was the only node left in the cluster. The other node instigated a Transfer Of Control (TOC) to preserve data integrity (split-brain syndrome) because it did not obtain the cluster lock. The application running on the node that instigated a Transfer Of Control (TOC) had to be restarted on the remaining node. The moral of the story is twofold:

i. Be very careful if you are going to run processes at or above priority = 20.
ii. If you are going to use high priority processes, consider increasing your `NODE_TIMEOUT`.

Below, we look at analyzing the resulting crashdump. We are interested in establishing a number of facts:

i. Check out the cluster configuration files and `syslog.log` for additional information.
ii. Was the crash a TOC instigated by Serviceguard?
iii. When was the last time `cmcld` ran?

In my example, I simply ran `STOP cmcld` by sending it a signal 24, i.e., `kill -STOP $(cat /var/adm/cmcluster/cmcld.pid)` on the machine `hpeos002`. This is obviously something I *do not* suggest that you undertake on a live system. Here's the output form `cmviewcl -v`:

```
root@hpeos001[cmcluster] # cmviewcl -v
CLUSTER      STATUS
McBond       up

  NODE          STATUS       STATE
  hpeos001      up           running

    Network_Parameters:
```

```
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      up           8/16/6       lan0
    STANDBY      up           8/20/5/2     lan1

  NODE          STATUS       STATE
  hpeos002      down         failed

    Network_Parameters:
    INTERFACE    STATUS       PATH         NAME
    PRIMARY      unknown      2/0/2        lan0
    STANDBY      unknown      4/0/1        lan1
root@hpeos001[cmcluster] #
```

We would follow this up by analyzing the information from `syslog.log`:

```
Aug  2 19:52:00 hpeos001 cmcld: Timed out node hpeos002. It may have failed.
Aug  2 19:52:00 hpeos001 cmcld: Attempting to adjust cluster membership
Aug  2 19:52:02 hpeos001 inetd[4426]: registrar/tcp: Connection from hpeos001
(192.168.0.201) at Fri Aug  2 19:52:02 2002
Aug  2 19:52:06 hpeos001 vmunix: SCSI: Reset requested from above -- lbolt:
547387, bus: 1
Aug  2 19:52:06 hpeos001 cmcld: Obtaining Cluster Lock
Aug  2 19:52:09 hpeos001 vmunix: SCSI: Resetting SCSI -- lbolt: 547687, bus: 1
Aug  2 19:52:09 hpeos001 vmunix: SCSI: Reset detected -- lbolt: 547687, bus: 1
Aug  2 19:52:16 hpeos001 cmcld: Unable to obtain Cluster Lock. Operation timed
out.
Aug  2 19:52:16 hpeos001 cmcld: WARNING: Cluster lock disk /dev/dsk/c1t0d0 has
failed.
Aug  2 19:52:16 hpeos001 cmcld: Until it is fixed, a single failure could
Aug  2 19:52:16 hpeos001 cmcld: cause all nodes in the cluster to crash
Aug  2 19:52:16 hpeos001 cmcld: Attempting to form a new cluster
Aug  2 19:52:23 hpeos001 cmcld: Obtaining Cluster Lock
Aug  2 19:52:24 hpeos001 cmcld: Cluster lock /dev/dsk/c1t0d0 is back on-line
Aug  2 19:52:24 hpeos001 cmcld: Turning off safety time protection since the
cluster
Aug  2 19:52:24 hpeos001 cmcld: may now consist of a single node. If Serviceguard
Aug  2 19:52:24 hpeos001 cmcld: fails, this node will not automatically halt
```

The "*SCSI: Reset – lbolt*" messages in this instance is as a result of the node resetting the SCSI interface after another node leaves the cluster. Should you see any "*SCSI: Reset – lbolt*" messages during normal operation, you should investigate them as a separate hardware-related problem.

You can see that I obtain the cluster lock after the SCSI reset. I am now the only member of the cluster.

Here's the crashdump analysis I performed on the resulting TOC of `hpeos002`. Input commands will be underlined. Interesting findings will be highlighted with a larger font and accompanying notes:

```
root@hpeos002[] # cd /var/adm/crash
root@hpeos002[crash] # ll
```

```
total 4
-rwxr-xr-x   1 root       root             1 Aug  2 21:01 bounds
drwxr-xr-x   2 root       root          1024 Aug  2 21:01 crash.0
root@hpeos002[crash] # cd crash.0
root@hpeos002[crash.0] # ll
total 129856
-rw-r--r--   1 root       root          1176 Aug  2 21:01 INDEX
-rw-r--r--   1 root       root       8372224 Aug  2 21:01 image.1.1
-rw-r--r--   1 root       root       8364032 Aug  2 21:01 image.1.2
-rw-r--r--   1 root       root       8368128 Aug  2 21:01 image.1.3
-rw-r--r--   1 root       root       8376320 Aug  2 21:01 image.1.4
-rw-r--r--   1 root       root       8388608 Aug  2 21:01 image.1.5
-rw-r--r--   1 root       root       4390912 Aug  2 21:01 image.1.6
-rw-r--r--   1 root       root       20223172 Aug  2 21:01 vmunix
root@hpeos002[crash.0] # more INDEX
comment   savecrash crash dump INDEX file
version   2
hostname  hpeos002
modelname 9000/715
panic     TOC, pcsq.pcoq = 0.15f4b0, isr.ior = 0.91fcf0
```

NOTE: Although this tells us the system instigated a Transfer Of Control (TOC), it doesn't tell us why.

```
dumptime  1028318274 Fri Aug   2 20:57:54 BST 2002
savetime  1028318469 Fri Aug   2 21:01:09 BST 2002
release   @(#)     $Revision: vmunix:    vw: -proj    selectors:
CUPI80_BL2000_1108 -c 'Vw for CUPI80_BL2000_1108 build' -- cupi80_bl2000_1108
'CUPI80_BL2000_1108'  Wed Nov  8 19:05:38 PST 2000 $
memsize   268435456
chunksize 8388608
module    /stand/vmunix vmunix 20223172 3848474440
image     image.1.1 0x0000000000000000 0x00000000007fc000 0x0000000000000000
0x0000000000001127 3658315572
image     image.1.2 0x0000000000000000 0x00000000007fa000 0x0000000000001128
0x00000000000019ef 2052742134
image     image.1.3 0x0000000000000000 0x00000000007fb000 0x00000000000019f0
0x00000000000030af 1656526062
image     image.1.4 0x0000000000000000 0x00000000007fd000 0x00000000000030b0
0x00000000000090bf 2888801859
image     image.1.5 0x0000000000000000 0x0000000000800000 0x00000000000090c0
0x000000000000c97f 1440262390
image     image.1.6 0x0000000000000000 0x0000000000430000 0x000000000000c980
0x000000000000ffff 320083218
root@hpeos002[crash.0] #
root@hpeos002[crash.0] # q4pxdb vmunix
.
Procedures: 13
Files: 6
root@hpeos002[crash.0] # q4 -p .
@(#) q4 $Revision: B.11.20f $ $Fri Aug 17 18:05:11 PDT 2001 0
Reading kernel symbols ...
```

```
Reading data types ...
Initialized PA-RISC 1.1 (no buddies) address translator ...
Initializing stack tracer ...
script /usr/contrib/Q4/lib/q4lib/sample.q4rc.pl
executable /usr/contrib/Q4/bin/perl
version 5.00502
SCRIPT_LIBRARY = /usr/contrib/Q4/lib/q4lib
perl will try to access scripts from directory
/usr/contrib/Q4/lib/q4lib

q4: (warning) No loadable modules were found
q4: (warning) No loadable modules were found
q4> ex &msgbuf+8 using s
NOTICE: nfs3_link(): File system was registered at index 3.
NOTICE: autofs_link(): File system was registered at index 6.
NOTICE: cachefs_link(): File system was registered at index 7.
1 graph3
2 bus_adapter
2/0/1 c720
2/0/1.0 tgt
2/0/1.0.0 sdisk
2/0/1.1 tgt
2/0/1.1.0 sdisk
2/0/1.3 tgt
2/0/1.3.0 stape
2/0/1.6 tgt
2/0/1.6.0 sdisk
2/0/1.7 tgt
2/0/1.7.0 sctl
2/0/2 lan2
2/0/4 asio0
2/0/6 CentIf
2/0/8 audio
2/0/10 fdc
2/0/11 ps2
5 bus_adapter
5/0/1 hil
5/0/2 asio0
4 eisa
4/0/1 lan2
8 processor
9 memory

    System Console is on the ITE
Entering cifs_init...
Initialization finished successfully... slot is 9
Logical volume 64, 0x3 configured as ROOT
Logical volume 64, 0x2 configured as SWAP
Logical volume 64, 0x2 configured as DUMP
    Swap device table:  (start & size given in 512-byte blocks)
        entry 0 - major is 64, minor is 0x2; start = 0, size = 1048576
    Dump device table:  (start & size given in 1-Kbyte blocks)
        entry 00000000 - major is 31, minor is 0x6000; start = 88928, size =
```

```
524288
Starting the STREAMS daemons-phase 1
Create STCP device files
         $Revision: vmunix:    vw: -proj    selectors: CUPI80_BL2000_1108 -c 'Vw
for CUPI80_BL2000_1108 build' -- cupi80_bl2000_1108 'CUPI80_BL2000_1108'  Wed
Nov  8 19:05:38 PST 2000 $
Memory Information:
    physical page size = 4096 bytes, logical page size = 4096 bytes
    Physical: 262144 Kbytes, lockable: 185460 Kbytes, available: 213788 Kbytes

NOTICE: vxvm:vxdmp: added disk array OTHER_DISKS, datype = OTHER_DISKS

Serviceguard: Unable to maintain contact with cmcld daemon.
Performing TOC to ensure data integrity.
```

NOTE: As we can see here, Serviceguard has given us a clear message that there is something wrong in the cluster. Let us find the `cmcld` process:

```
q4> load struct proc from proc_list max nproc next p_global_proc
loaded 134 struct procs as a linked list (stopped by null pointer)
q4> print p_pid p_uid p_comm | grep cmcld
 3791     0   "cmcld"
q4> keep p_pid==3791
kept 1 of 134 struct proc's, discarded 133
q4> load struct kthread from p_firstthreadp
loaded 1 struct kthread as an array (stopped by max count)
q4> trace pile
stack trace for process at 0x0`0316a040 (pid 3791), thread at 0x0`02f5e800 (tid
3897)
process was not running on any processor
_swtch+0xc4
_sleep+0x2f4
select+0x5e4
syscall+0x6ec
$syscallrtn+0x0
q4> print kt_tid kt_pri kt_lastrun_time ticks_since_boot
  3897    532          104635           110504
```

NOTE: Here, we can see the priority of `cmcld` = 20. The internal, kernel priorities are offset by 512 to the external user priorities. We can also see the discrepancy between the last time this thread ran and the cumulative ticks (10 milliseconds) since the system was booted.

```
q4>
q4> (ticks_since_boot - kt_lastrun_time)/100
072     58      0x3a
```

NOTE: As we can see, 58 (decimal) seconds passed since `cmcld` ran. This is some time outside of our `NODE_TIMEOUT`, so we can conclude that we are now into the time when the

election would be running and this node, having lost the election, instigated a Transfer Of Control (TOC).

```
q4> history
HIST NAME   LAYOUT COUNT TYPE           COMMENTS
   1 <none>   list   134 struct proc    stopped by null pointer
   2 <none> mixed?    1 struct proc    subset of 1
   3 <none>  array     1 struct kthread stopped by max count
q4> recall 2
copied a pile
q4> print p_pid p_uid p_stat p_cursig p_comm
p_pid p_uid p_stat p_cursig  p_comm
 3791     0  SSTOP       24 "cmcld"
q4>
```

NOTE: We sent signal 24 (`STOP` signal) and hence `p_cursig` is set. This is confirmed by the STATE of the process = `SSTOP`.

Process priorities do not constitute the only reason why `cmcld` may be starved for resources. It could be due to many other reasons. The idea here is for us to attempt to establish any possible reasons why we experienced a cluster reformation and resulting TOC. Even if we are confident about the reasons surrounding the cluster reformation and resulting TOC, I would **strongly** suggest that you place a Software Support Call to your local Hewlett-Packard Response Center to have a trained Response Center Engineer analyze the crashdump in detail and come to his own conclusions. We can pass on our initial findings in order to try to speed up the process of finding a root cause for this problem. It is always wise to have professional, experienced help in establishing the root cause of any system crash. Your local Response Center will continue with the investigation to establish the root cause of the failure.

We now know that our cluster has formed successfully and is behaving "*properly*." We have ensured that certain critical tests have shown that Serviceguard can accommodate local LAN failures easily and it reacts, as expected, when critical components, i.e., `cmcld`, becomes starved for resources. At this time, we can conclude our discussions on a package-less cluster. I know of a number of installations that use Serviceguard simply to provide automatic LAN failover, i.e., what we have demonstrated so far. If that is all you want Serviceguard to perform, then that's it … happy clustering. However, most of us will be using Serviceguard to protect our applications from the types of failures we considered earlier. After all, it is the applications that run on these systems that give these systems their value within the organization. Let's move on to consider Packages. We look at constructing from "scratch" and consider using some of the various Serviceguard Toolkits.

25.8 Constant Monitoring

Now that our cluster is up and running, it needs constant and careful monitoring. In Chapter 10, "Monitoring System Resources," we discussed EMS HA Monitors. EMS HA Monitors can be configured to monitor the state of hardware resources as well as critical system resources. A

Serviceguard cluster can be considered a critical system resource. As such, a cluster can be monitored from within the EMS framework to send alarms to users and/or applications such as OpenView networking monitoring tools if the cluster experiences an expected event such as a LAN card or even an entire node failing.

In Chapter 27, we look at using the Serviceguard Manager GUI to monitor cluster resources. Whatever tool you use to monitor clusters, you should review all your IT processes in order to take into account the requirements of a High Availability Cluster.

■ Chapter Review

Serviceguard is a tool that forms part of a complete High Availability solution. Serviceguard does not offer any form of fault tolerance. Serviceguard was designed to minimize the downtime involved in having applications processes run on a different node. If a critical resource fails, Serviceguard can automate the process of having applications run on an adoptive node. This will involve a degree of downtime for the application while the necessary checks are made in order to start up the application on another node. This is the main topic of discussion in the next chapter.

In this chapter, we looked at a simple two-node cluster. Large clusters (up to 16 nodes) offer greater flexibility when dealing with a failure. As we see in upcoming chapters, Serviceguard can be configured with a measure of intelligence when moving applications to *adoptive nodes*; Serviceguard will choose the adoptive node that is currently running the smallest number of applications.

Serviceguard is available for HP-UX as well as Linux (where the concept of Quorum Servers was born). The configuration of Serviceguard on Linux is the same as on HP-UX, offering seamless migration between the two platforms.

Our cluster is currently not monitoring the status of any applications. This is a feature of Serviceguard that most administrators will want to investigate. This is the topic we discuss next.

▲ Test Your Knowledge

1. *It is important that all nodes in the cluster have the same IO tree for shared devices. If not, we have to get involved with recreating the Instance numbers assigned to devices via reconfiguring the* `/etc/ioconfig` *and* `/stand/ioconfig` *files. True or False?*

2. *Every* `HEARTBEAT_INTERVAL` *node is transmitting heartbeat packets. After two subsequent* `HEARTBEAT_INTERVALS` *where the heartbeat packet does not reach the cluster coordinator, a cluster reformation commences. True or False?*

3. *Which of the following events will trigger a cluster reformation? Select all the correct answers.*

   A. A node leaves the cluster.

   B. An application is moved to an adoptive node.

   C. An application fails on its current node.

   D. A node joins the cluster.

   E. A Primary LAN interface fails on the Cluster Coordinator node.

   F. The cluster starts up automatically.

   G. The cluster starts up manually.

4. *A network interface designated as a `STATIONARY_IP` interface will not have its IP configuration moved to a Standby LAN interface in the event of a LAN card failure. True or False?*

5. *To create the cluster binary file, we use the `cmapplyconf` command. We need 100 percent attendance when the cluster binary file is first created. The first time we start the cluster, we don't need 100 percent node attendance as long as all nodes have the newly created cluster binary file. True or False?*

▲ Answers to Test Your Knowledge

1. *False. Nodes can use their own device file naming convention for shared devices, as long as the devices are referenced correctly in the cluster configuration file.*

2. *False. After a `NODE_TIMEOUT` interval, a cluster reformation commences.*

3. *Answers A, D, F, and G are correct.*

4. *False. A `STATIONARY_IP` interface simply means that no heartbeat packets are transmitted over that particular interface.*

5. *False. We need 100 percent node attendance the first time the cluster is started.*

▲ Chapter Review Questions

1. *What is the difference between an Active/Standby and a Rolling/Standby cluster configuration?*

2. *The disk/volume groups that are going to be shared between nodes in the cluster necessitate a different series of standard configuration files that normally deal with and manage disk/volumes/filesystems. Which standard configuration files are affected and why?*

3. Looking at the following cluster ASCII configuration file, make any comments on the validity of this configuration:

```
# **********************************************************************
# ********* HIGH AVAILABILITY CLUSTER CONFIGURATION FILE ***************
# ***** For complete details about cluster parameters and how to    ****
# ***** set them, consult the Serviceguard manual. ****
# **********************************************************************

# Enter a name for this cluster. This name will be used to identify the
# cluster when viewing or manipulating it.

CLUSTER_NAME            finance


# Cluster Lock Parameters
#
# The cluster lock is used as a tiebreaker for situations
# in which a running cluster fails, and then two equal-sized
# sub-clusters are both trying to form a new cluster. The
# cluster lock may be configured using either a lock disk
# or a quorum server.
#
# You can use either the quorum server or the lock disk as
# a cluster lock but not both in the same cluster.
#
# Consider the following when configuring a cluster.
# For a two-node cluster, you must use a cluster lock. For
# a cluster of three or four nodes, a cluster lock is strongly
# recommended. For a cluster of more than four nodes, a
# cluster lock is recommended. If you decide to configure
# a lock for a cluster of more than four nodes, it must be
# a quorum server.

# Lock Disk Parameters. Use the FIRST_CLUSTER_LOCK_VG and
# FIRST_CLUSTER_LOCK_PV parameters to define a lock disk.
# The FIRST_CLUSTER_LOCK_VG is the LVM volume group that
# holds the cluster lock. This volume group should not be
# used by any other cluster as a cluster lock device.

# Quorum Server Parameters. Use the QS_HOST, QS_POLLING_INTERVAL,
# and QS_TIMEOUT_EXTENSION parameters to define a quorum server.
# The QS_HOST is the host name or IP address of the system
# that is running the quorum server process. The
# QS_POLLING_INTERVAL (microseconds) is the interval at which
# Serviceguard checks to make sure the quorum server is running.
# The optional QS_TIMEOUT_EXTENSION (microseconds) is used to increase
# the time interval after which the quorum server is marked DOWN.
#
# The default quorum server timeout is calculated from the
# Serviceguard cluster parameters, including NODE_TIMEOUT and
# HEARTBEAT_INTERVAL. If you are experiencing quorum server
```

```
# timeouts, you can adjust these parameters, or you can include
# the QS_TIMEOUT_EXTENSION parameter.
#
# For example, to configure a quorum server running on node
# "qshost" with 120 seconds for the QS_POLLING_INTERVAL and to
# add 2 seconds to the system assigned value for the quorum server
# timeout, enter:
#
# QS_HOST qshost
# QS_POLLING_INTERVAL 120000000
# QS_TIMEOUT_EXTENSION 2000000

FIRST_CLUSTER_LOCK_VG           /dev/vg01


# Definition of nodes in the cluster.
# Repeat node definitions as necessary for additional nodes.

NODE_NAME               fin01
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP                192.1.1.1
  NETWORK_INTERFACE     lan1
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c1t0d0
# List of serial device file names
# For example:
SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

NODE_NAME               fin02
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP                192.1.1.2
  NETWORK_INTERFACE     lan1
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c0t0d0
# List of serial device file names
# For example:
SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

NODE_NAME               fin03
  NETWORK_INTERFACE     lan0
    HEARTBEAT_IP                192.1.1.3
  NETWORK_INTERFACE     lan1
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c0t0d0
# List of serial device file names
# For example:
SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

NODE_NAME               fin04
  NETWORK_INTERFACE     lan0
```

```
    HEARTBEAT_IP                192.1.1.4
  NETWORK_INTERFACE     lan1
  FIRST_CLUSTER_LOCK_PV /dev/dsk/c0t0d0
# List of serial device file names
# For example:
SERIAL_DEVICE_FILE    /dev/tty0p0

# Possible standby Network Interfaces for lan0: lan1.

# Cluster Timing Parameters (microseconds).
# The NODE_TIMEOUT parameter defaults to 2000000 (2 seconds).
# This default setting yields the fastest cluster reformations.
# However, the use of the default value increases the potential
# for spurious reformations due to momentary system hangs or
# network load spikes.
# For a significant portion of installations, a setting of
# 5000000 to 8000000 (5 to 8 seconds) is more appropriate.
# The maximum value recommended for NODE_TIMEOUT is 30000000
# (30 seconds).

HEARTBEAT_INTERVAL              6000000
NODE_TIMEOUT            5000000

# Configuration/Reconfiguration Timing Parameters (microseconds).

AUTO_START_TIMEOUT      600000000
NETWORK_POLLING_INTERVAL        2000000

# Package Configuration Parameters.
# Enter the maximum number of packages which will be configured in the
# cluster.
# You can not add packages beyond this limit.
# This parameter is required.
MAX_CONFIGURED_PACKAGES         10

# List of cluster aware LVM Volume Groups. These volume groups will
# be used by package applications via the vgchange -a e command.
# Neither CVM or VxVM Disk Groups should be used here.
# For example:
# VOLUME_GROUP          /dev/vgdatabase
# VOLUME_GROUP          /dev/vg02

VOLUME_GROUP            /dev/vg01
```

4. *Explain why the Cluster Management Daemon (`cmcld`) is run at an HP-UX Real-Time Priority of 20. Does this have any implications on how we manage our own processes/applications?*

5. *Where is the configuration parameter `AUTOSTART_CMCLD` stored? What is its default value? What does this parameter control? Give at least three reasons why you would set the configuration parameter `AUTOSTART_CMCLD=0`.*

▲ ANSWERS TO CHAPTER REVIEW QUESTIONS

1. *With an Active/Standby configuration, a node is designated as a Standby node and runs an application only when a failure occurs. If the same node is deemed a Standby node for multiple applications, it may have to run multiple applications in the event of multiple failures. A Rolling/Standby configuration is where there is an initial Standby node ready to take over in the event of a failure. The difference is that every node that sustains a failure can subsequently become a Standby node; in this way, the responsibility of being a Standby node rolls over to the node that is currently not running an application.*

2. *Two files are affected:*

   A. `/etc/lmrc`: This startup script needs to be modified to *not* activate all volume groups at startup time. Serviceguard will activate volume groups as necessary.

   B. `/etc/fstab`: Any filesystems that will be shared between nodes in the cluster must *not* be listed in `/etc/fstab because` Serviceguard will mount any filesystems when starting up associated applications.

3. *The cluster has four nodes. The following points can be made regarding the configuration:*

   A. Using a serial heartbeat in a four-node cluster is not supported.

   B. Using a cluster-lock disk in a four-node cluster is supported but *unusual.*

   C. The `HEARTBEAT_INTERVAL` = 6 seconds. The `NODE_TIMEOUT` = 5 seconds. As such, the cluster will be constantly reforming because the nodes will be timing out before sending heartbeat packets.

   The cluster configuration is invalid.

4. *An HP-UX Real-Time Priority of 20 is a very high priority and gives a process a high probability of being executed when it needs to. The `cmcld` process is the most important process in the cluster because it coordinates the sending and receiving of heartbeat packets. If this process cannot run, the node will not be able to send/receive heartbeat packets and will be deemed to have failed. This will cause a cluster reformation, and the node in question may end up instigating a Transfer Of Control (TOC). The implications for managing our own processes/applications is that if we run processes at a priority of 20 or greater, there is a possibility that the `cmcld` process will not be allowed to execute and will cause Serviceguard to instigate a Transfer Of Control (TOC) because application processes are monopolizing the processors.*

5. *`AUTOSTART_CMCLD` is stored in the startup configuration file `/etc/rc.config.d/cmcluster`. Its default value is 0. The parameter controls where the node will attempt to rejoin the cluster after the system is rebooted. There are three reasons to set the parameter to = 0.*

A. The cluster will normally be started the first time by the `cmruncl` command. Once the cluster is up and running, the nodes in the cluster should remain up as long as possible. If a node is rebooted, it must be for a reason. If a node does not rejoin the cluster automatically, i.e., `AUTOSTART_CMCLD`=0, it can indicate to the administrator(s) that something unexpected has happened to the node.

B. If a node is experiencing hardware/software problems that cause it to reboot repeatedly and `AUTOSTART_CMCLD`=1, the node would be attempting to rejoin the cluster a number of times. This will cause a cluster reformation that can potentially mask other problems with individual nodes or the cluster as a whole.

C. When a node is started up after some hardware/software maintenance, it is often the case that an administrator will want to ensure that any hardware/software updates/changes have been effective before allowing the node to rejoin the cluster. Having `AUTOSTART_CMCLD`=0 will allow the system to be rebooted as normal without attempting to rejoin the cluster, allowing the administrator to perform any additional configuration checks as necessary.