Redbooks Paper

Dino Quintero
Shawn Bodily
Patrick Pothier
Octavian Lascu

HACMP V5.3, Dynamic LPAR, and Virtualization

Introduction / Overview

This IBM® Redpaper is a practical case study about HACMP™ V5.3, Dynamic LPAR, and Advanced POWER™ Virtualization on pSeries® servers, including recommendations on how to deploy and configure HACMP in this environment.

This Redpaper focuses on the latest LPAR management techniques for IBM @server® pSeries servers, including implementation techniques, installation changes, and system management, in addition to HACMP V5.3 features, to provide a highly available environment.

This Redpaper is targeted at HACMP V5.3 technical professionals (consultants, IT architects, and IT specialists) responsible for delivering solutions involving High Availability on pSeries servers.

In this Redpaper, the following topics are discussed:

- Implementing HACMP and dynamic LPAR (on POWER4™ systems)
- Implementing HACMP and virtualization (vio/vscsi/vlan)

The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Dino Quintero is a Senior Certified Consulting IT Specialist for the ITSO in Poughkeepsie, New York. Before joining ITSO, he worked as a Performance Analyst for the Enterprise Systems Group and as a Disaster Recovery Architect for IBM Global Services. His areas of expertise include disaster recovery and pSeries clustering solutions. He is certified on pSeries system administration and pSeries clustering technologies. He is also an IBM Senior Certified Professional on pSeries technologies. Currently, he leads technical teams delivering Redbook solutions on pSeries clustering technologies and delivering technical workshops worldwide.

Shawn Bodily is a Senior I/T Specialist for ATS Americas in Dallas, Texas. He has worked for IBM for seven years now. He has ten years of AIX® experience and seven years of specializing in HACMP. He is HACMP and ATE certified for both V4 and V5. He has written and presented on high availability and storage. This is his second redbook.

Patrick Pothier has been an IT specialist in FTSS pSeries for STG France for five years. He has 13 years experience in the UNIX® field and 11 years experience in HACMP. His areas of experience include operating systems (MVS™, AIX, Solaris™, and Linux®), high availability (HACMP), backup solutions (ADSM/6000, Tivoli® Storage Manager, and Netbackup), and ERP administration (SAP R/3 BC).

Octavian Lascu is a Project Leader at the International Technical Support Organization, Poughkeepsie Center. He writes extensively and teaches IBM classes worldwide on all areas of pSeries and Linux clusters. Before joining the ITSO, Octavian worked in IBM Global Services Romania as a software and hardware Services Manager. He holds a Master's Degree in Electronic Engineering from the Polytechnical Institute in Bucharest and is also an IBM Certified Advanced Technical Expert in AIX/PSSP/HACMP. He has worked with IBM since 1992.

Thanks to the following people for their contributions to this project:

Wade Wallace
International Technical Support Organization, Austin Center

Mike Coffey, Paul Moyer, Elaine Krakower
IBM Poughkeepsie

Tom Weaver
IBM Austin

David Truong
IBM Dallas

Implementing dynamic LPAR with HACMP

Since this an advanced topic, a basic understanding of LPAR, dynamic LPAR, and virtualization and their associated terminology is assumed. Detailed information about these topics can be found in the redbooks:

- *Partitioning Implementations for IBM @server p5 Servers,* SG24-7039
- *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration,* SG24-7940

These redbooks, and many others, can be downloaded from:

`http://www.redbooks.ibm.com/`

The information and steps provided are an addition to an existing HACMP cluster. The steps of configuring an entire HACMP cluster are omitted to reduce replication of steps in this publication.

In this section, we cover the following topics in regards to dynamic LPAR:

- Requirements
- Application provisioning
- Defining dynamic LPAR to HACMP
- Our test configuration
- Test results
- Additional considerations

We expect that proper LPAR and dynamic LPAR planning is part of the overall process before implementing any similar configuration. It is important to understand not only the requirements, but to also understand the overall effects each decision has on the overall implementation.

Requirements

To use the integrated dynamic LPAR functions or Capacity Upgrade on Demand (CUoD) function of HACMP on POWER4, all LPAR nodes in the cluster should have at *least* the following levels installed:

- AIX 5L™ V5.2
- HACMP V5.2.0.1 (via IY58557 for dynamic LPAR)
- APAR IY58497 (for CUoD support)
- RSCT V2.3.3.1
- OpenSSH V3.4p1

APARs are required for support of CUoD and dynamic LPAR, as follows:

- CUoD on POWER5 systems:
  - For HACMP 5.3: APAR IY73051
  - For HACMP 5.2: APAR IY69525
- For HACMP 5.2 with CUoD on POWER4 systems: APAR IY58497 and IY65930
- For HACMP 5.3 with dynamic LPAR on POWER5 systems: APAR IY73050
- For HACMP 5.2 with dynamic LPAR on POWER4 systems: APAR IY58557
- For HACMP 5.2 with dynamic LPAR on POWER5 systems: APAR IY74493 ==> currently only available via ifix.

The OpenSSH software can be obtained from any of the following sources:

- AIX 5L V5.2 Bonus pack
- AIX 5L V5.3 Expansion pack
- Linux Toolbox CD

  This software can be downloaded from:

  `http://sourceforge.net/projects/openssh-aix`

OpenSSh for AIX has its own software prerequisites:

- rpm.rte
- zlib compression/decompression library (zlib)
- Pseudo random number generator daemon (prngd)
- OpenSSL Cryptographic Libraries (OpenSSL)

These prerequisites can be downloaded from:

`http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html`

The OpenSSL package can be found by clicking the **AIX Toolbox Cryptographic Content** link, registering, and accepting the license agreement.

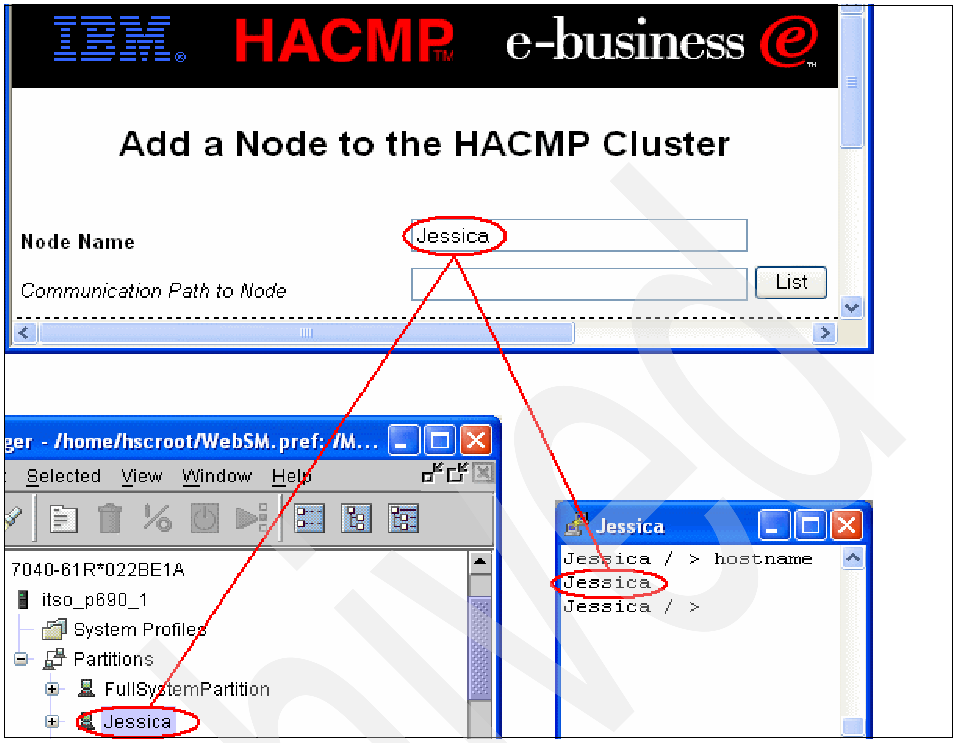
HMC attachment to the LPARs is required for proper management and dynamic LPAR capabilities. The HMC must be network attached on a network that it has in common with the LPARs in order to allow remote dynamic LPAR operations. The HMC must also have at *least* the following levels installed:

- HMC 3 V2.6
- HMC build level/firmware 20040113.1 or later

Important: APAR IY69525 for HACMP V5.2 or APAR IY73051 for HACMP V5.3 is required to support dynamic LPAR, CUoD, and CBU functionality on POWER5™ systems.

At the time of the writing of this Redpaper, the APARs required for POWER5 support were unavailable. Any additional software levels needed for POWER5 support have yet to be determined.

Attention: A key configuration requirement is that the LPAR partition name, the AIX host name, and the HACMP node name must all match. We show this in Figure 1 on page 6.


IBM HACMP e-business
Add a Node to the HACMP Cluster
Node Name
Communication Path to Node
List
ger - /home/hscroot/WebSM.pref: /M...
Selected View Window Help
7040-61R*022BE1A
itso_p690_1
System Profiles
Partitions
FullSystemPartition
Jessica

```
Jessica / > hostname
Jessica
Jessica / >
```

Figure 1   Partition name, AIX host name, and HACMP node name matching

Other considerations

There are several things to consider when planing a cluster to include dynamic LPAR operations. These include but are not limited to:

- Encountering possible config_too_long during dynamic LPAR events
- Mix of LPARs and non-LPAR systems
- CUoD provisioning

As POWER5 dynamic LPAR/CUoD support becomes available, there are additional possible configurations:

- Mixing POWER4 and POWER5 dynamic LPAR
- Using shared or dedicated CPUs
- Using capped or uncapped CPUs

As with any cluster, the configuration must be tested thoroughly. This includes anything that can be done to simulate or produce a real work load for the most realistic test scenarios as possible.

Application provisioning

Details on this topic can be found *High Availability Cluster Multi-Processing for AIX Administration Guide Version 5.3*, SC23-4862.

This section describes the flow of actions in the HACMP cluster, if the application provisioning function through dynamic LPAR and CUoD is configured. It also includes several examples that illustrate how resources are allocated, depending on different resource requirements.

Overview

When you configure an LPAR on the HMC (outside of HACMP), you provide the minimum, desired, and maximum values for the number of CPUs and amount of memory for the LPAR. These values can be obtained by running the **`lshwres`** command on the HMC. The stated minimum values of the resources must be available when an LPAR node starts. If more resources are available in the free pool on the frame, an LPAR can allocate up to the stated desired values.

During dynamic allocation operations, the system does not allow the values for the CPU and memory to go below the minimum or above the maximum amounts specified for the LPAR.

HACMP obtains the LPAR minimum and maximum amounts and uses them to allocate and release CPU and memory when application servers are started and stopped on the LPAR node.

HACMP requests the dynamic LPAR resource allocation from the HMC before the application servers are started, and releases the resources after the application servers are stopped. The Cluster Manager waits for the completion of these events before continuing the event processing in the cluster.

HACMP handles the resource allocation and release for application servers serially, regardless if the resource groups are processed in parallel. This minimizes conflicts between application servers trying to allocate or release the same CPU or memory resources. Therefore, you must carefully configure the cluster to properly handle all CPU and memory requests on an LPAR.

These considerations are important:

- Once HACMP has acquired additional resources for the application server, when the application server moves again to another node, HACMP releases only those resources that are no longer necessary to support this application on the node.
- HACMP does *not* start and stop LPAR nodes.

It is possible to create a custom event or customize application start/stop scripts to stop LPAR nodes if desired.

Acquiring dynamic LPAR and CUoD resources

If you configure an application server that requires a minimum and desired amount of resources (CPU or memory), HACMP determines if additional resources need to be allocated for the node and allocates them if possible.

In general, HACMP tries to allocate as many resources as possible to meet the desired amount for the application, and uses CUoD, if allowed, to do this.

The LPAR node has the LPAR minimum

If the node owns only the minimum amount of resources, HACMP requests additional resources through dynamic LPAR and CUoD (if applicable).

In general, HACMP starts counting the extra resources required for the application from the minimum amount. That is, the minimum resources are retained for the node's overhead operations, and are *not* utilized to host an application.

The LPAR node has enough resources to host an application

The LPAR node that is about to host an application may already contain enough resources (in addition to the LPAR minimum) to meet the desired amount of resources for this application.

In this case, HACMP does not allocate any additional resources and the application can be successfully started on the LPAR node. HACMP also calculates that the node has enough resources for this application in addition to hosting all other application servers that may be currently running on the node.

Resources requested from the free pool and from the CUoD pool

If the amount of resources in the free pool is insufficient to satisfy the total amount requested for allocation (the minimum requirements for one or more applications), HACMP requests resources from CUoD (if enabled).

If HACMP meets the requirement for a minimum amount of resources for the application server, application server processing continues. Application server processing continues even if the total desired resources (for one or more applications) have not been met or are only partially met. In general, HACMP attempts to acquire up to the desired amount of resources requested for an application.

If the amount of resources is insufficient to host an application, HACMP starts resource group recovery actions to move the resource group to another node.

Minimum amount requested for an application cannot be satisfied

In some cases, even after HACMP requests to use resources from the CUoD pool, the amount of resources it can allocate is less than the minimum amount specified for an application.

If the amount of resources is still insufficient to host an application, HACMP starts resource group recovery actions to move the resource group to another node.

The LPAR node is hosting application servers

In all cases, HACMP checks whether the node is already hosting application servers that required application provisioning, and that the LPAR maximum for the node is not exceeded:

- Upon subsequent fallovers, HACMP checks if the minimum amount of requested resources for yet another application server plus the amount of resources already allocated to applications residing on the node exceeds the LPAR maximum.
- In this case, HACMP attempts resource group recovery actions to move the resource group to another LPAR. Note that when you configure the dynamic LPAR and CUoD requirements for this application server, then during cluster verification, HACMP warns you if the total number of resources requested for all applications exceeds the LPAR maximum.

Allocation of resources in a cluster with multiple applications

If you have multiple applications in different resource groups in the cluster with LPAR nodes, and more than one application is configured to potentially request additional resources through the dynamic LPAR and CUoD function, the resource allocation in the cluster becomes more complex.

Based on the resource group processing order, some resource groups (hence the applications) might not be started. We explain this further in “Examples of using dynamic LPAR and CUoD resources” on page 11

Releasing dynamic LPAR and CUoD resources

When the application server is stopped on the LPAR node (the resource group moves to another node), HACMP releases only those resources that are no longer necessary to support this application server on the node. The resources are released to the free pool on the frame.

HACMP first releases the dynamic LPAR or CUoD resources it acquired last. This implies that the CUoD resources may not always be released before the dynamic LPAR resources are released.

The free pool is limited to the single frame only. That is, for clusters configured on two frames, HACMP does not request resources from the second frame for an LPAR node residing on the first frame.

Also, if LPAR 1 releases an application that puts some dynamic LPAR resources into the free pool, LPAR 2, which is using the CUoD resources, does not make any attempt to release its CUoD resources and acquire the free dynamic LPAR resources.

Stopping LPAR nodes

When the Cluster Manager is forced down on an LPAR node, and that LPAR is subsequently shutdown (outside of HACMP), the CPU and memory resources are released (not by HACMP) and become available for other resource groups running on other LPARs. HACMP does not track CPU and memory resources that were allocated to the LPAR and does not retain them for use when the LPAR node rejoins the cluster.

Note: If you are using the On/Off license for CUoD resources, and the LPAR node is shutdown (outside of HACMP), the CUoD resources are released (not by HACMP) to the free pool, but the On/Off license continues to be turned on. You may need to manually turn off the licence for the CUoD resources that are now in the free pool. (This ensures that you do not pay for resources that are not being currently used.)

If the LPAR is not stopped after the Cluster Manager is forced down on the node, the CPU and memory resources remain allocated to the LPAR for use when the LPAR rejoins the cluster.

Changing the dynamic LPAR and CUoD resources dynamically

You can change the dynamic LPAR and CUoD resource requirements for application servers without stopping the cluster services. Synchronize the cluster after making the changes.

The new configuration is not reflected until the next event that causes the application (hence the resource group) to be released and reacquired on another node. In other words, a change in the resource requirements for CPUs, memory, or both does not cause the recalculation of the dynamic LPAR resources. HACMP does not stop and restart application servers solely for the purpose of making the application provisioning changes.

If *another* dynamic reconfiguration change (that is, rg_move) causes the resource groups to be released and reacquired, the new resource requirements for dynamic LPAR and CUoD are used at the end of this dynamic reconfiguration event.

Examples of using dynamic LPAR and CUoD resources

The following examples explain CPU allocation and release. The process for memory is very similar. While these are descriptions of how it works, we also provide real results from our test configuration in "Test results" on page 31.

Note: Be aware that once HACMP acquires additional resources for an application server, when the server moves again to another node, it takes the resources with it, that is, the LPAR node releases all the additional resources it acquired.

The configuration is an eight CPU frame, with a two-node (each an LPAR) cluster. There are two CPUs available in the CUoD pool through CUoD activations. The nodes have the partition profile characteristics shown in Table 1 and Table 2 on page 12.

Table 1 Profile characteristics

Node name	LPAR minimum	LPAR maximum
Longhorn	1	9
Hurricane	1	5

There are three application servers defined, each belonging to separate resource groups.

Table 2 Application requirements

Application server name	CPU desired	CPU minimum	Allow CUoD
App1	1	1	Yes
App2	2	2	No
App3	4	4	No

Example 1: No CPUs added at start, some are released upon stop

The starting configuration settings are as follows:

- Longhorn has three CPUs allocated.
- Hurricane has one CPU allocated.
- Free pool has four CPUs allocated.

The applications servers are started in the following order:

- Longhorn starts App2, and no CPUs are allocated to meet the requirement of three CPUs. (Three CPUs is equal to the sum on Node1's LPAR minimum of one plus App2's desired amount of two).
- Longhorn stops App2. Two CPUs are released, leaving one CPU, which meets the minimum requirement. (Since no other application servers are running, the only requirement is Longhorn's LPAR minimum of one).

Example 2: No CPUs added due to the resource group processing order

In this example, we start off with the same configuration settings given in "Example 1: No CPUs added at start, some are released upon stop" on page 12.

The application servers are started as follows:

- Longhorn starts App1, and no CPUs are allocated since the requirement of two is met. Longhorn starts App3, and three CPUs are allocated to meet the requirement of six. There is now one CPU in the free pool.
- Longhorn attempts to start App2. After Longhorn has acquired App1 and App3, the total amount of CPUs Longhorn must now own to satisfy these requirements is six, which is the sum of Longhorn's LPAR minimum of one, App1's desired amount of one, and App3's desired amount of four.

Since App2's minimum amount is two, in order to acquire App2, Longhorn needs to allocate two more CPUs, but there is only one CPU left in the free pool and it does not meet the minimum requirement of two CPUs for App2. The resource group with App2 is not acquired locally, as there is only one CPU in the free pool and CUoD use is not allowed. If no other member nodes are present, then the resource group goes into the error state.

If node hurricane would have been a member node of the App 2 resource group, and active in the cluster, then an rg_move would have been invoked in an attempt to bring up the resource group on node hurricane.

Example 3: Successful CUoD resources allocation and release

The starting configuration settings are as follows:

- Longhorn has three CPUs allocated.
- Hurricane has one CPU allocated.
- The free pool has four CPUs.

The application servers are started in the following order:

- Longhorn starts App3, and two CPUs are allocated to meet the requirement of five.
- Longhorn starts App2, and two CPUs are allocated to meet the requirement of seven. There are now no CPUs in the free pool.
- Longhorn starts App1, and one CPU is taken from CUoD and allocated to meet the requirement of eight.
- Longhorn stops App3, and four CPUs are released and one of those CPUs is put back into the CUoD pool.

Example 4: Resource group failure, minimum resources not met

In this example, the resource group acquisition fails due to the fact that the minimum resources needed are not currently available, as the LPAR has reached its maximum.

The configuration is as follows:

- Longhorn has one CPU.
- Hurricane has one CPU.
- Free pool has six CPUs.

The application servers are started in the following order:

- Hurricane starts App3, and four CPUs are allocated to meet the requirement of five. There are now two CPUs in the free pool.

- Hurricane attempts to start App2, but App2 goes into an error state because the LPAR maximum for hurricane is five and hurricane cannot acquire more CPUs.

> **Note:** If the minimum resources for App2 would have been set to zero, the acquisition would have succeeded, as no additional resources would have been required.

Example 5: Resource group failure, LPAR min and max are the same

In this example, we demonstrate a real example we encountered during our early testing. This is a direct result of improper planning in regards to how application provisioning works.

We are still using an eight CPU frame, but the additional application servers and nodes are not relevant to this example. The LPAR configuration for node longhorn is shown in Table 3.

Table 3 LPAR characteristics for node longhorn

LPAR minimum	LPAR desired	LPAR maximum
4	4	4

The App1 application server has the settings shown in Table 4.

Table 4 Application requirements for App1

Minimum number of CPUs	Desired number of CPUs
1	4

The starting configuration is as follows:

- Longhorn has four CPUs allocated.
- Free pool has four CPUs.

The App1 application server is started locally on node longhorn. During acquisition, the LPAR minimum is checked and added to the application server minimum, which returns a total of five. This total exceeds the LPAR maximum setting and results in the resource group going into the error state.

Though the LPAR may technically already have enough resources to host the application, because of the combination of settings, it results in a failure. Generally speaking, you would not have equal minimum and maximum settings.

This scenario could have been avoided in one of three ways:

- Change the LPAR minimum to three or less.
- Change the LPAR maximum to more than four.
- Change App1's minimum number of CPUs to zero.

Configuring dynamic LPAR in HACMP

We will cover the following steps that are needed to configure dynamic LPAR in an HACMP cluster:

- Name resolution
- Install ssh on HACMP nodes
- Configure HMC ssh access
- Define HMC and managed systems to HACMP
- Define dynamic LPAR resources to HACMP (that is, application provisioning)

Name resolution

One common issue seen is name resolution not being consistent across all systems. If name resolution is not configured correctly, the dynamic LPAR feature cannot be used. The underlying Reliable Scalable Cluster Technology (RSCT) infrastructure expects an identical host name resolution on all participating nodes. If this is not the case, RSCT will be unable to communicate properly.

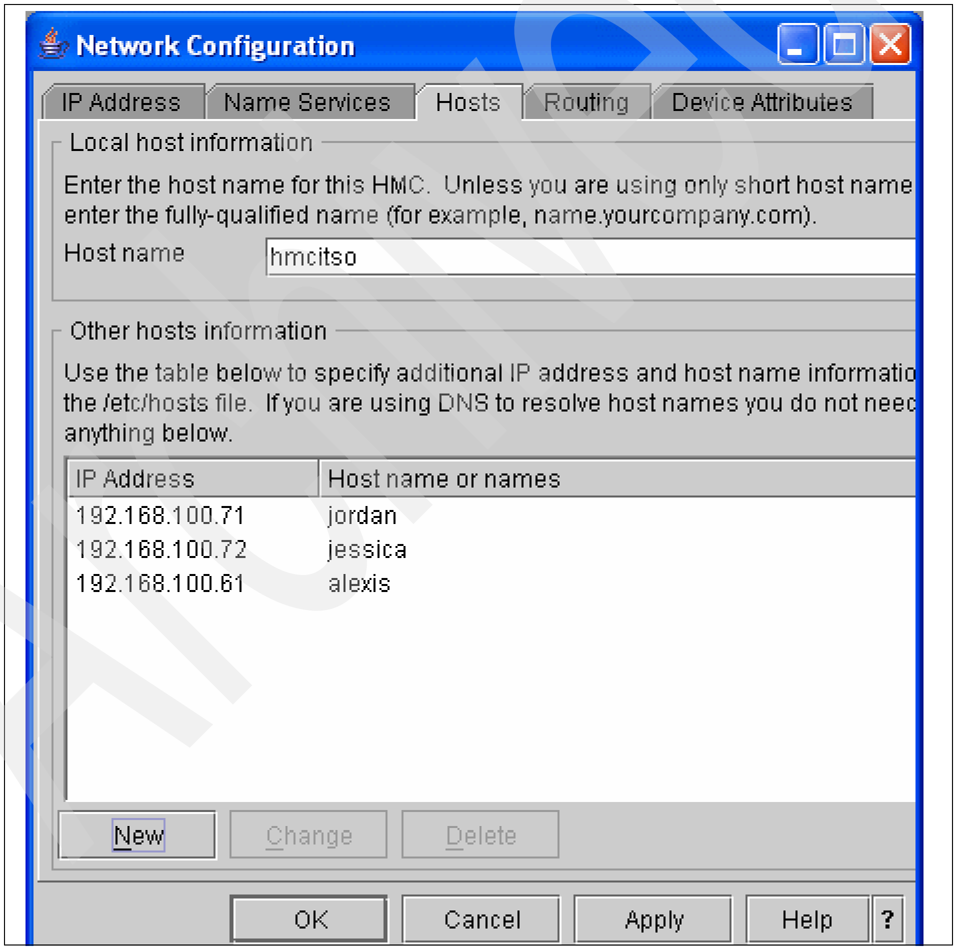
Ensure that all nodes and the HMC are configured identically by checking the following list. The phrase "All systems" includes all HACMP nodes and the HMC.

- All systems must resolve the participating host names and IP addresses identically. This includes reverse name resolution.
- All systems must use the same type of name resolution, either short or long name resolution.
- All systems should use the same name resolution order, either local or remote. To ensure this, check the following files:
  - /etc/hosts on all systems
  - /etc/netsvc.conf on all AIX nodes
  - /etc/host.conf on the HMC

We assume that you know how to check these files on the AIX systems. However, checking these files on the HMC may not be as commonly known.

Make sure that the HMC is aware of the host LPARs. The host names and IPs should be listed in the HMC hosts list. We recommend configuring the hosts information through the HMC console since each version of the HMC code continues to restrict command line options. You can verify this on the POWER4 HMC by selecting **HMC Maintenance** → **System Configuration** → **Customize Network Settings** → **Hosts**. If the addresses and names are not listed, you can add them by clicking on **New**. Our HMC host file configuration is shown in Figure 2.

Note: If using POWER5 HMCs, click **HMC Management** → **HMC Configuration** → **Customize Network Settings**.



Local host information

Enter the host name for this HMC. Unless you are using only short host name enter the fully-qualified name (for example, name.yourcompany.com).

Host name: hmcitso

Other hosts information

Use the table below to specify additional IP address and host name informatio the /etc/hosts file. If you are using DNS to resolve host names you do not neec anything below.

IP Address	Host name or names
192.168.100.71	jordan
192.168.100.72	jessica
192.168.100.61	alexis

Figure 2 HMC hosts

Install and configure SSH on HACMP nodes

In order to use remote command operations on the HMC, you are required to have SSH installed on the HACMP nodes. The HMC must be configured to allow access from these partitions.

In this section, we cover installing the ssh packages, including what order to install them in. These packages have very little flexibility in their install order. In most cases, if something is installed out of order, an error message will normally point you to which package is needed first.

With each version of SSH and HMC code, these steps may differ slightly. We have documented the processes we used to successfully implement our environment. Information for installing SSH on each version of AIX can be found at:

`http://www-1.ibm.com/support/docview.wss?uid=isg1pTechnote0707`

Installing SSH

In "Requirements" on page 4, we covered which packages are needed and where to obtain them. The following steps assume that these packages have been downloaded or copied onto the HACMP nodes. We chose to put all of our images in the common install directory of /usr/sys/inst.images.

The rpm.rte package must be installed prior to installing the additional rpm packages. You can use either **`installp`** or **`smit install_all`** to install it. You can verify it is installed by running **`lslpp -l rpm.rte`** (see Example 1).

Example 1 Checking if rpm is installed

```
Jordan / > lslpp -l rpm.rte
  Fileset                      Level  State      Description
  ----------------------------------------------------------------------------
Path: /usr/lib/objrepos
  rpm.rte                   3.0.5.36  COMMITTED  RPM Package Manager
Path: /etc/objrepos
  rpm.rte                   3.0.5.36  COMMITTED  RPM Package Manager
```

It is now possible to install the remaining prerequisites by using the **`rpm`** command. We installed these packages as shown in Example 2.

Example 2 Installing openSSH prerequisites

```
rpm -i zlib-1.2.1-2.aix5.1.ppc.rpm
rpm -i prngd-0.9.23-3.aix4.3.ppc.rpm
rpm -i openssl-0.9.7d-2.aix5.1.ppc.rpm
rpm -i openssl-devel-0.9.7d-2.aix5.1.ppc.rpm
rpm -i openssl-doc-0.9.7d-2.aix5.1.ppc.rpm
```

This fulfills the requirements needed to install SSH. In AIX 5L V5.1 and above, the openSSH package is in **installp** format. Assuming the image has been extracted from the tar package, you can now install it using **smitty install_all**. There are three core filesets to install. These filesets and the results of our install are shown in Example 3.

Example 3 Install SSH

```
smitty install_all
> openssh.base                                                           ALL ¦
  ¦    + 3.8.0.5202  Open Secure Shell Commands                              ¦

  ¦    + 3.8.0.5202  Open Secure Shell Server                                ¦

  ¦ > openssh.license                                                    ALL ¦

  ¦    + 3.8.0.5202  Open Secure Shell License                               ¦

  ¦                                                                          ¦

  ¦ > openssh.man.en_US                                                  ALL ¦

  ¦    + 3.8.0.5202  Open Secure Shell Documentation - U.S. English          ¦


Name                    Level           Part        Event       Result

-------------------------------------------------------------------------

openssh.license         3.8.0.5202      USR         APPLY       SUCCESS

openssh.base.client     3.8.0.5202      USR         APPLY       SUCCESS

openssh.base.server     3.8.0.5202      USR         APPLY       SUCCESS

openssh.base.client     3.8.0.5202      ROOT        APPLY       SUCCESS

openssh.base.server     3.8.0.5202      ROOT        APPLY       SUCCESS
openssh.man.en_US       3.8.0.5202      USR         APPLY       SUCCESS
```

Tip: Be sure to choose yes on the field to accept the license agreement.

Now that SSH is installed, we need to configure the HACMP nodes to access the HMC without passwords for remote dynamic LPAR operations.

Configure HMC SSH access

The document *Managing the Hardware Management Console (HMC)* contains a section that describes the steps to set up a remote secure shell access. The guide can be downloaded from the IBM Web site at:

http://publib.boulder.ibm.com/infocenter/iseries/v1r2s/en_US/info/iphai/iphai.pdf

These are the steps we used in our setup to enable SSH access from our HACMP nodes

1. Enable HMC SSH access.
2. Generate SSH keys on HACMP nodes.
3. Enable non-password HMC access via the authorized_keys2 file.

First, make sure the HMC is set up to allow remote operations by doing the following:

1. In the Navigation area, select **HMC Maintenance**.
2. In the Navigation area, select **System Configuration**.
3. In the Contents area, click **Enable/Disable Remote Command Execution**.
4. Select the box to enable ssh.

Note: Normally, we recommend creating a separate HMC user for remote command execution; however, HACMP uses hscroot.

We need to create the SSH directory $HOME/.ssh for the root user to store the authentication keys. HACMP will execute the ssh remote dynamic LPAR operations as the root user. By default, this is /.ssh, which is what we used.

To generate public and private keys, run the following command on each HACMP node:

```
/usr/bin/ssh-keygen -t rsa
```

This will create the following files in /.ssh:

```
private key: id_rsa
public key: id_rsa.pub
```

The write bits for both group and other are turned off. Ensure that the private key has a permission of 600.

The HMC's public key needs to be in the known_hosts file on each HACMP node, and vice versa. This is easily accomplished by executing **ssh** to the HMC from each HACMP node. The first time **ssh** is executed, a prompt will be displayed to insert the key into the file. Answer yes to continue, and then you will be prompted to enter a password. It is necessary to do so, as we have not allowed non-password ssh access yet (see Example 4).

Example 4   SSH to HMC

```
Jordan /tmp > ssh -l hscroot 192.168.100.69

The authenticity of host '192.168.100.69 (192.168.100.69)' can't be
established.

 RSA key fingerprint is 2d:50:3f:03:d3:51:96:27:5a:5e:94:f4:e3:9b:e7:78

 Are you sure you want to continue connecting (yes/no)?yes
Warning: Permanently added '192.168.100.69' (RSA) to the list of known
hosts.
```

When utilizing two HMCs, like in our test configuration, it is necessary to repeat this process for each HMC. You may also wish to do this between all member nodes to allow ssh type of operations between them (that is, scp, sftp, and ssh).

To allow non-password ssh access, we must put each HACMP node's public key into the authorized_keys2 file on the HMC. This can be done more than one way; however, here is an overview of the steps we used:

1. Create the authorized_keys2 file on the HMC.
2. Copy (use **scp**) the public key from all HACMP nodes to one of the nodes.
3. Concatenate (use **cat**) all the key files together into the authorized_keys2 file.
4. Copy (use **scp**) the concatenated file over to the HMC /home/hscrtoot/.ssh.

For the first step, we created the authorized_keys2 file manually. This can be done by either using the command line at the HMC or remotely from the client. We executed it remotely from the client. We chose to create a dummy file first, then copy (using **scp**) the contents of our combined key files over to the HMC, essentially replacing our dummy file. To create the file with dummy information, we ran, on one client:

```
ssh hscroot@hmc “mkauthkeys --add ‘*’ ”
```

Verify on the HMC that the authorized_keys2 file exists in the .ssh directory. We did this from the client by executing:

```
ssh hscroot@hmc "ls -al .ssh/"
```

Note: You can run the **mkauthkeys** command via ssh from each AIX client and add one key at time, as documented in the *Managing the Hardware Management Console (HMC)*. However, syntactically, it requires adding the key string in manually. We found this to be cumbersome, especially when having to execute on multiple system.

Next, from /.ssh on the AIX LPARs, we made a copy of the public key and renamed it to include the local node name as part of the file name. We then copied, via **scp**, the public key of each machine (Jessica and Alexis) to one node (Jordan). We then ran the **cat** command to create an authorized_keys2 file that contains the public key information for all HACMP nodes. The commands run on each node are shown in Example 5.

Example 5 Copying the authorized_keys2 file to HMC

```
Alexis /.ssh > cp id_rsa.pub id_rsa.pub.alexis
Alexis /.ssh > scp id_rsa.pub.alexis jordan:/.ssh/id_rsa.pub.alexis

Jessica /.ssh > cp id_rsa.pub id_rsa.pub.jessica
Jessica /.ssh > scp id_rsa.pub.jessica jordan:/.ssh/id_rsa.pub.jessica

Jordan /.ssh > cp id_rsa.pub id_rsa.pub.jordan
Jordan /.ssh > cat id_rsa.pub.alexis id_rsa.pub.jessica id_rsa.pub.jordan
>authorized_keys2

Jordan /.ssh > ls -al
total 64
drwx------   2 root     system         256 Jul 18 22:27 .
drwxr-xr-x  21 root     system        4096 Jul 14 02:11 ..
-rw-r--r--   1 root     system         664 Jun 16 16:31 authorized.keys2
-rw-------   1 root     system         883 Jun 16 14:12 id_rsa
-rw-r--r--   1 root     system         221 Jun 16 14:12 id_rsa.pub
-rw-r--r--   1 root     system         221 Jun 16 16:30 id_rsa.pub.alexis
-rw-r--r--   1 root     system         222 Jun 16 15:20 id_rsa.pub.jessica
-rw-r--r--   1 root     system         221 Jun 16 16:27 id_rsa.pub.jordan
-rw-r--r--   1 root     system        1795 Jul 14 04:08 known_hosts

Jordan/.ssh > scp authorized.keys2 hscroot@192.168.100.69:.ssh/authorized_keys2
hscroot@192.168.100.69's password:
authorized_keys2                              100% 664 0.7KB/s   00:00
```

When executing **scp** on the HMC, you should be prompted to enter the password for the hscroot user. Once entered, authorized_key2 will be copied. You can then test if the no-password access is working from each node by executing the **ssh** command, as shown in Example 4 on page 20. However, this time you should end up at the HMC shell prompt, as shown in Example 6 on page 22.

Example 6 Test no-password ssh access

```
Alexis /.ssh > ssh -l hscroot 192.168.100.69
Last login:Thur Jun 16 22:46:51 2005 from 192.168.100.61
hscroot@hmcitso:~>
```

Once each node can **ssh** to the HMC without a password, then this step is completed and HACMP verification of the HMC communications should succeed.

Defining HMC and managed system names

The HMC's IP addresses must be specified for each HACMP node that will be

utilizing dynamic LPAR. In our example, each HACMP node corresponds to an LPAR. Each LPAR is assigned to a *managed system* (CEC). Managed systems are those systems that are physically attached to and managed by the same HMC. These managed systems must also be defined to HACMP.

You can obtain the managed system names through the HMC console in the navigation area. The managed system name can be a user created name or the default name (the machine type and serial number).

To define the HMC communication for each HACMP node:

1. Run **smit hacmp** and select Extended Configuration → Extended Resource Configuration → HACMP Extended Resources Configuration → Configure HACMP Applications → Configure HACMP for Dynamic LPAR and CUoD Resources → Configure Communication Path to HMC → Add HMC IP Address for a Node and press Enter. The Add HMC IP Address panel appears.

Tip: You can also use the SMIT fast path of **smit cladd_apphmc.dialog** to accomplish step 1.

2. Fill out the following fields as appropriate:

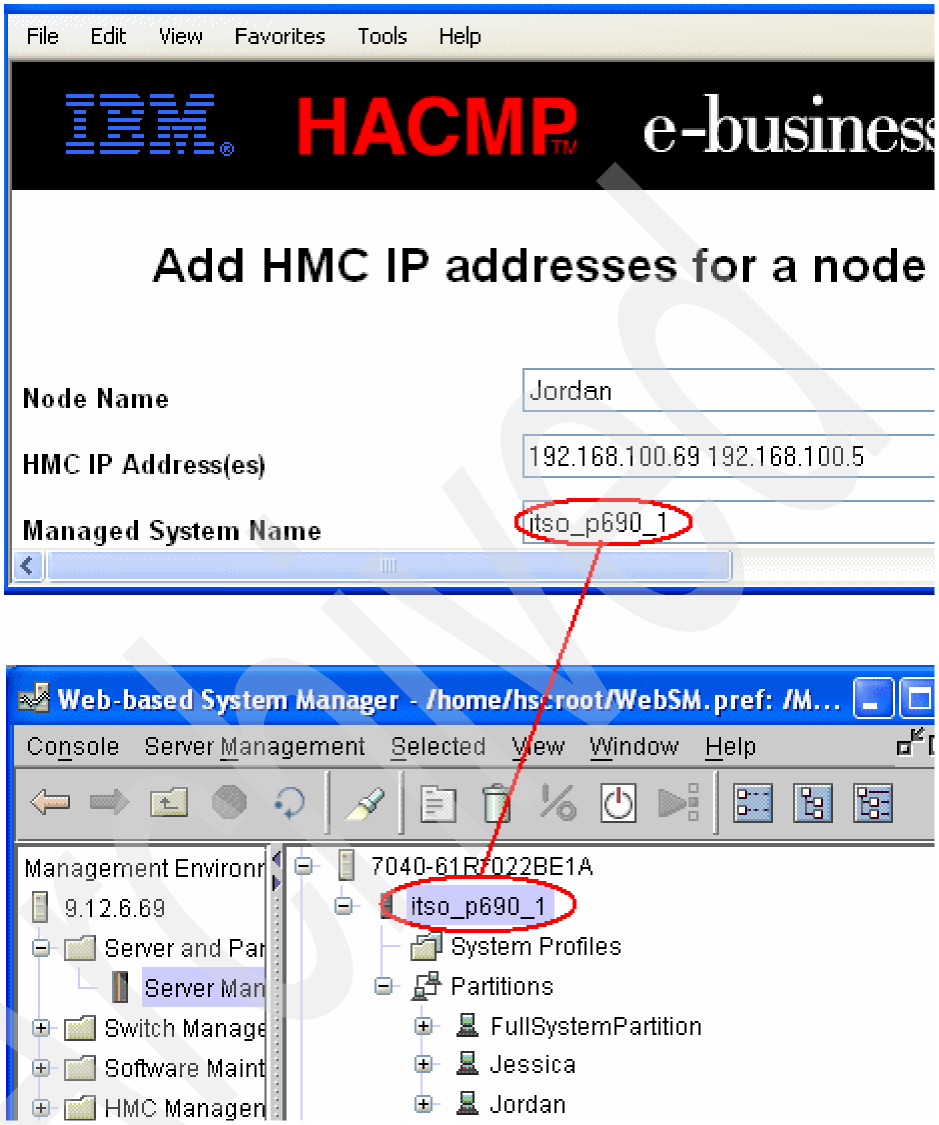
Node Name	Select a node name to associate with one or more Hardware Management Console (HMC) IP addresses and a Managed System.
HMC IP Address(es)	Enter one or more space-separated IP addresses for the HMC. If addresses are added for more than one HMC, HACMP tries to communicate with each HMC until a working communication path is found. Once the communication path is established, HACMP uses this path to execute the dynamic logical partition commands on that HMC.

Managed System Name

Enter the name of the Managed System that runs the LPAR that represents the node. The maximum length is 32 characters.

3. Press Enter.

Figure 3 on page 24 shows the smit screen to which we added the HMC information. We also show the HMC managed system name information used in our test configuration.


File Edit View Favorites Tools Help

IBM HACMP e-business

Add HMC IP addresses for a node

Node Name	Jordan
HMC IP Address(es)	192.168.100.69 192.168.100.5
Managed System Name	itso_p690_1

Web-based System Manager - /home/hscroot/WebSM.pref: /M...

Console Server Management Selected View Window Help

Management Environm
9.12.6.69
Server and Par
Server Man
Switch Manage
Software Maint
HMC Managem

7040-61R*022BE1A
itso_p690_1
System Profiles
Partitions
FullSystemPartition
Jessica
Jordan

Figure 3 Defining HMC and Managed System to HACMP

Note: Appendix E, "Using dynamic LPAR and CUoD in an HACMP Cluster", in the *High Availability Cluster Multi-Processing for AIX Administration Guide*, SC23-4862 states that the managed system name cannot include underscores. However, in our example, we did use underscores and it worked fine. Discussions with development reaffirmed that this should not be a stated limitation.

During cluster verification, HACMP verifies that the HMC is reachable by first issuing a **`ping`** to the IP address specified. If the HMC responds, then HACMP will verify that each specified HACMP node is in fact dynamic LPAR capable by issuing an **`lssycfg`** command, via ssh, on the HMC.

Configure application provisioning

To configure dynamic LPAR and CUoD resources, for each application server that could use dynamic LPAR-allocated or CUoD resources, do the following steps:

1. Run **`smit hacmp`** and select Extended Configuration → Extended Resource Configuration → HACMP Extended Resources Configuration → Configure HACMP Applications → Configure HACMP for Dynamic LPAR and CUoD Resources → Configure Dynamic LPAR and CUoD Resources for Applications → Add Dynamic LPAR and CUoD Resources for Applications and press Enter.

   A pick list of configured application servers appears.

Tip: You can also use the SMIT fast path of **`smit cladd_appdlpar.dialog`** to accomplish step 1.

2. Select an application server from the list and press Enter.

   The panel to specify the requirements for an application server appears. Detailed information can be found in the help panels and in "Application provisioning" on page 7.

3. Fill out the following fields as appropriate:

   Application Server Name
   This is the application server for which you will configure Dynamic LPAR and CUoD resource provisioning that was chosen from the previous menu.

   Minimum Number of CPUs
   Enter the minimum number of CPUs to acquire when the application server starts. The default value is zero.

To perform the application provisioning, HACMP checks how many CPUs the LPAR node currently has above its LPAR minimum value, compares this number with the minimum requested in this field, and based on this, requests more CPUs, if needed.

Number of CPUs Enter the maximum amount of CPUs HACMP will attempt to allocate to the node before starting this application on this node. The default value is zero.

Minimum Amount of Memory
Enter the amount of memory to acquire when the application server starts. Must be a multiple of 256.

Use CUoD if resources are insufficient?
The default is No. Select Yes to have HACMP use CUoD to obtain enough resources to fulfill the minimum amount requested. Using CUoD requires a license key (activation code) to be entered on the HMC and may result in extra costs due to usage of the CUoD license.

I agree to use CUoD resources
The default is No. Select Yes to acknowledge that you understand that there might be extra costs involved when using CUoD. HACMP logs the answer to the syslog and smit.log files.

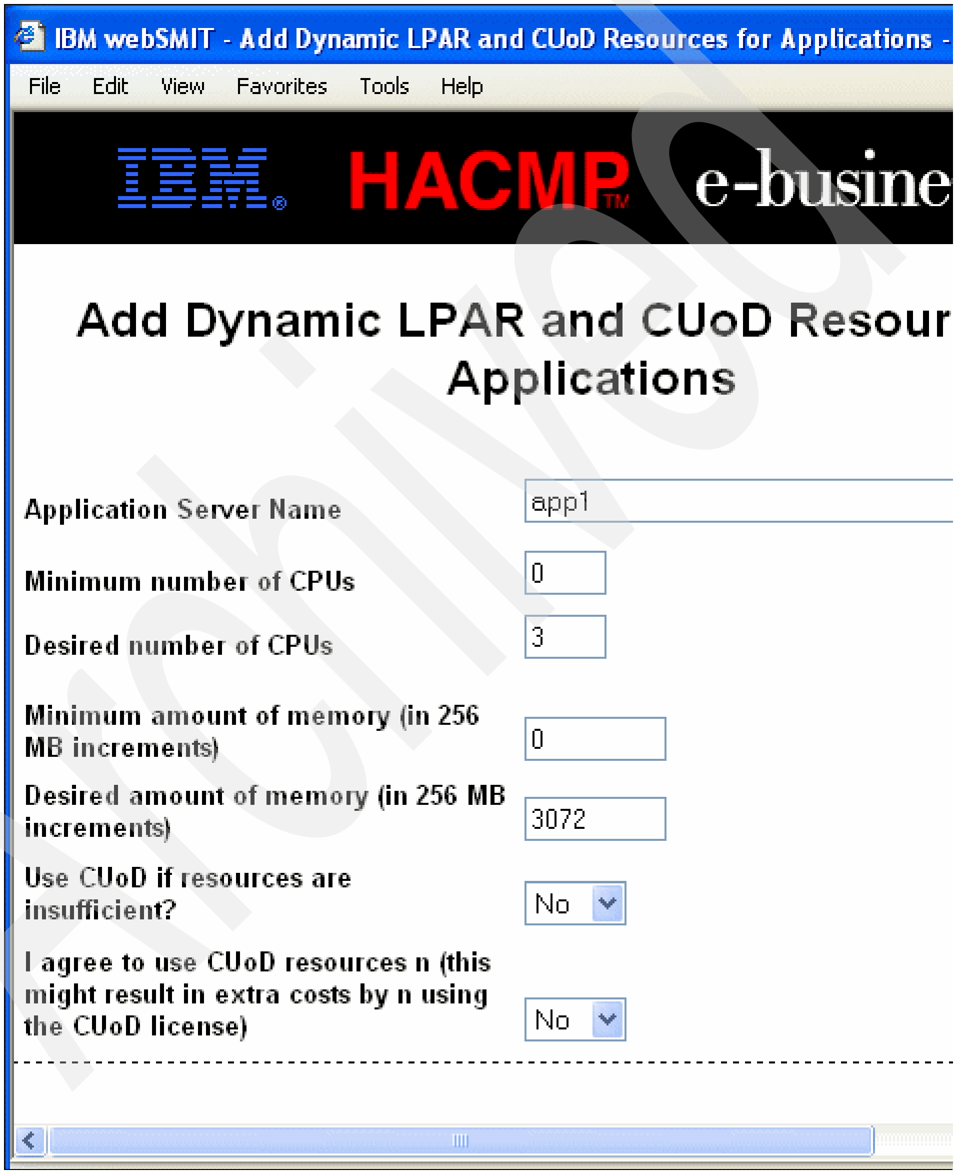
4. Press Enter.

When the application requires additional resources to be allocated on this node, HACMP performs its calculations to see whether it needs to request only the dynamic LPAR resources from the free pool on the frame and whether that would already satisfy the requirement, or if CUoD resources are also needed for the application server. After that, HACMP requests the desired amounts of memory and numbers of CPU, if you decided to use them.

During verification, HACMP ensures that the entered values are below LPAR maximum values for memory and CPU. Otherwise, HACMP issues an error, stating these requirements.

HACMP also verifies that the total of required resources for *all* application servers that can run concurrently on the LPAR is less than the LPAR maximum. If this requirement is not met, HACMP issues a warning. Note that this scenario can happen upon subsequent fallovers, that is, if the LPAR node is already hosting application servers that require dynamic LPAR and CUoD resources, then upon acquiring yet another application server, it is possible that the LPAR cannot acquire any additional resources beyond its LPAR maximum. HACMP verifies this case and issues a warning.

An application provisioning example for our test configuration can be seen in Figure 4.


IBM webSMIT - Add Dynamic LPAR and CUoD Resources for Applications -
File Edit View Favorites Tools Help
IBM HACMP e-busine

Add Dynamic LPAR and CUoD Resour Applications

Application Server Name	app1
Minimum number of CPUs	0
Desired number of CPUs	3
Minimum amount of memory (in 256 MB increments)	0
Desired amount of memory (in 256 MB increments)	3072
Use CUoD if resources are insufficient?	No
I agree to use CUoD resources n (this might result in extra costs by n using the CUoD license)	No

Figure 4 Adding application provision to HACMP

After adding both the HMC communications and application provisioning, a cluster synchronization is required.

Troubleshooting HMC verification errors

In this section, we show some errors that could be encountered during verification, along with possibilities of why the errors are generated. Though some of the error messages seem self explanatory, we believe any tips in troubleshooting are normally welcome.

In Example 7, the error message itself gives you a good idea of what caused the problem. Here are some things you can do to discover the source of the problem:

- Ping the HMC IP address.
- Manually ssh to the HMC via **`ssh -l hscroot@"HMC_IP_address"`**.

Example 7 HMC unreachable during verification

```
ERROR: The HMC with IP label 192.168.100.69 configured on node jordan is not
reachable. Make sure the HMC IP address is correct, the HMC is turned on and
connected to the network, and the HMC has OpenSSH installed and setup with the
public key of node jordan.
```

If the ssh is unsuccessful or prompts for a password, that indicates that ssh has not been properly configured.

If the message shown in Example 8 appears by itself, this is normally an indication that access to the HMC is working; however, the particular node's matching LPAR definition is not reporting that it is dynamic LPAR capable. You can verify this manually from the HMC command line, as shown in Example 9.

Example 8 Node not dynamic LPAR capable verification error

```
ERROR: An HMC has been configured for node jordan, but the node does
not appear to be DLPAR capable.
```

Example 9 Verify LPAR is dynamic LPAR capable

```
hscroot@hmcitso:~> lssyscfg -r lpar -m itso_p690_1 -n Jordan
Name     id  DLPAR  State    Profile       OpPanel
Jordan   001 NO     Running  Jordan_Prod
```

Note: The HMC command syntax can vary by HMC code levels and type.

This may be caused by something as simple as the node not running at least AIX 5L V5.2, which is required for dynamic LPAR operations. It can also be caused by delays in updating the RMC data structure.

During our testing, we ran several events within very short periods of time. At some point, we would see that our LPAR would report it was no longer dynamic LPAR capable. Then, after a short period, it would report back normally again. We believe this was due to the RMC information between the LPARs and the HMC being unsynchronized.

Test cluster configuration

Our test configuration consists of three LPARs distributed over two p690s. There are two production LPARs (Jordan and Jessica) in one p690 (itso_p690_1) and one standby LPAR (Alexis) in the second p690 (itso_p690_2). Each p690 contains eight CPUs, 8 GB of memory, and are attached to two HMCs for redundancy.

Each partition has the following software levels installed:

- AIX 5L V5.2 ML5
- HACMP V5.2.0.3
- RSCT V2.3.5.0
- rpm-3.0.5-37.aix5.1
- OpenSSH V3.8.1p1 (and the following prerequisites)
  - zlib 1.2.1-2
  - prngd 0.9.23-3
  - openssl 0.9.7d-2

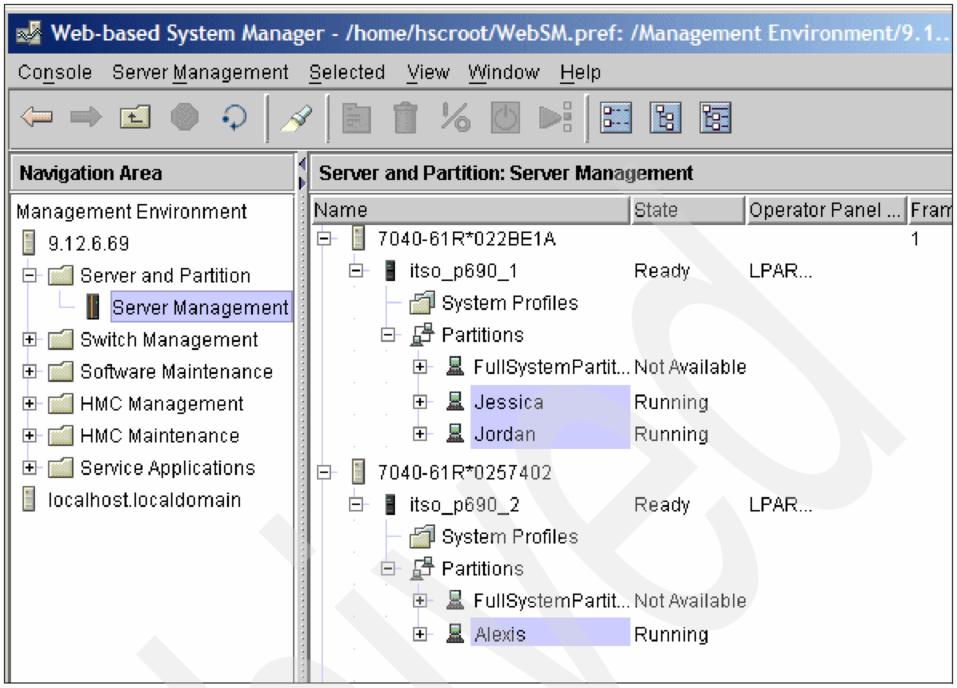
Each partition has the following adapters installed:

- 2 IBM Gigabit FC Adapters (#6228)
- 2 10/100 Ethernet adapters (#2975)

Both HMCs have:

- HMC 3 V3.5
- HMC build level/firmware 20050320.1

Our test LPARs and their corresponding frames can be seen in Figure 5 on page 30.


Web-based System Manager - /home/hscroot/WebSM.pref: /Management Environment/9.1..
Console
Server Management
Selected
View
Window
Help
Navigation Area
Server and Partition: Server Management

```
Management Environment
  9.12.6.69
    Server and Partition
      Server Management
    Switch Management
    Software Maintenance
    HMC Management
    HMC Maintenance
    Service Applications
  localhost.localdomain
```

```
Name                              State          Operator Panel ...   Fram
7040-61R*022BE1A                                                      1
  itso_p690_1                     Ready          LPAR...
    System Profiles
    Partitions
      FullSystemPartit...         Not Available
      Jessica                     Running
      Jordan                      Running
7040-61R*0257402
  itso_p690_2                     Ready          LPAR...
    System Profiles
    Partitions
      FullSystemPartit...         Not Available
      Alexis                      Running
```

Figure 5 Test LPARs

For shared storage, we used an ESS with eight 10 GB LUNs, four of which were assigned to each production resource group. For our testing purposes, the shared storage was not important, other than trying to set up a more complete cluster configuration by utilizing disk heartbeat.

These LPARs are configured with the partition profile settings shown in Table 5.

Table 5 Partition profile settings

LPAR name	Minimum	Desired	Maximum
Jordan	1 CPU - 1 GB	1 CPU - 1 GB	4 CPU - 4 GB
Jessica	1 CPU - 1 GB	1 CPU - 1 GB	2 CPU - 2 GB
Alexis	1 CPU - 1 GB	1 CPU - 1 GB	6 CPU - 6 GB

We have two resource groups configured, app1_rg and app2_rg, each containing their own corresponding application servers of app1 and app2, respectively. Each resource group is configured as online on a home node. App1_rg has the participating nodes of Jordan and Alexis. App2_rg has the participating nodes of Jessica and Alexis. This makes our cluster a 2+1 setup, with node Alexis as the standby node.

We have configured HMC communications for each node to include both HMCs with IP addresses of 192.168.100.69 and 192.168.100.5.

The application server dynamic LPAR configuration settings are shown in Table 6.

Table 6 Application server dynamic LPAR settings

App server	Minimum	Desired
app1	0	3
app2	0	2

We specifically chose the minimum settings of zero in order to always allow our resource group to be acquired.

Test results

The following paragraphs present the test results we obtained in our scenarios.

Scenario 1: Resource group acquisition

In this scenario, we start off with the following setup:

- *Jordan has one CPU/one GB allocated.*
- *The free pool has seven CPUs and 7 GB.*

Upon starting the cluster services on Jordan, app1 is started locally and attempts to acquire the desired amount of resources assigned. Since there are enough resources in the free pool, another three CPUs and 3 GB are acquired, as shown in Figure 6 on page 32.

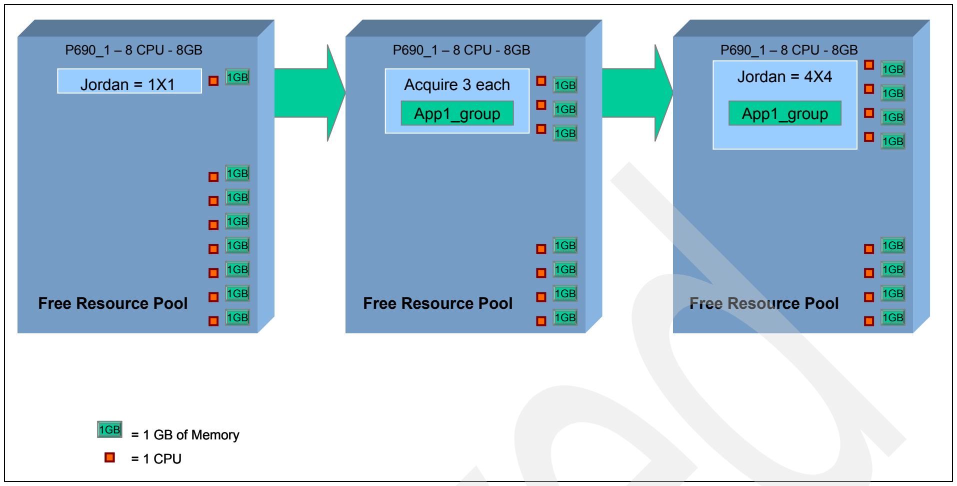


Figure 6   Dynamic LPAR resource acquisition

Scenario 2: Resource group release

In this scenario, node Jessica is online in the cluster with app2 running on its partition, with the maximum settings of two CPUs and 2 GB.

Upon stopping cluster services on Jessica, app2 is stopped locally and releases resources back to the free pool. When releasing resources, HACMP will not release more resources than it originally acquired.

In Figure 7, we show the releasing of resources and their re-allocation back to the free pool.

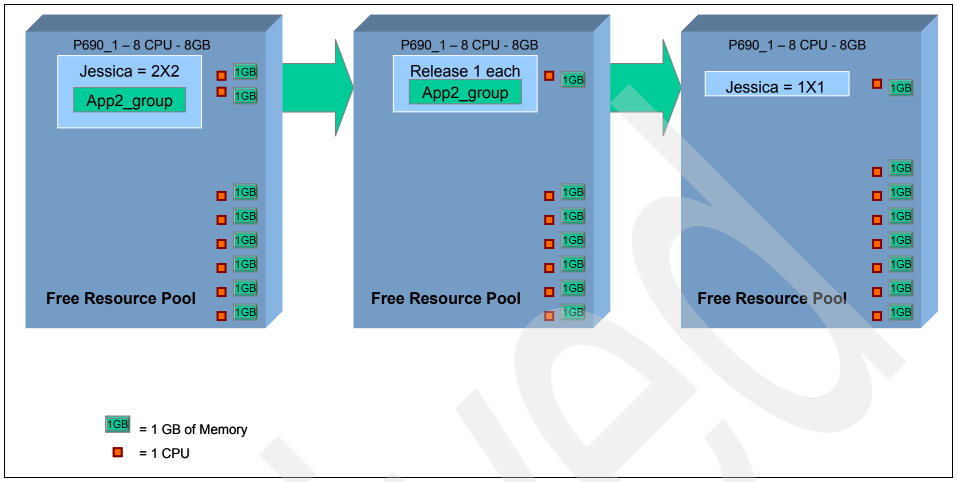


Figure 7 Dynamic LPAR resource release

Scenario 3: Fallover each LPAR sequentially

In this two part scenario, we will outline the process of falling over each partition, node Jordan, and then node Jessica. This demonstrates how resources are acquired on fallover, which is similar to local resource group acquisition.

Using this scenario and “Scenario 4: Fallover production LPARs in reverse order” on page 35, we show how each individual fallover differs in the total amount of resources the standby node uses.

For the first part, we fail node Jordan by executing **`reboot -q`**. This causes a fallover to occur to node Alexis. Alexis acquires the app1 resource group and allocates the desired amount of resources, as shown in Figure 8 on page 34.

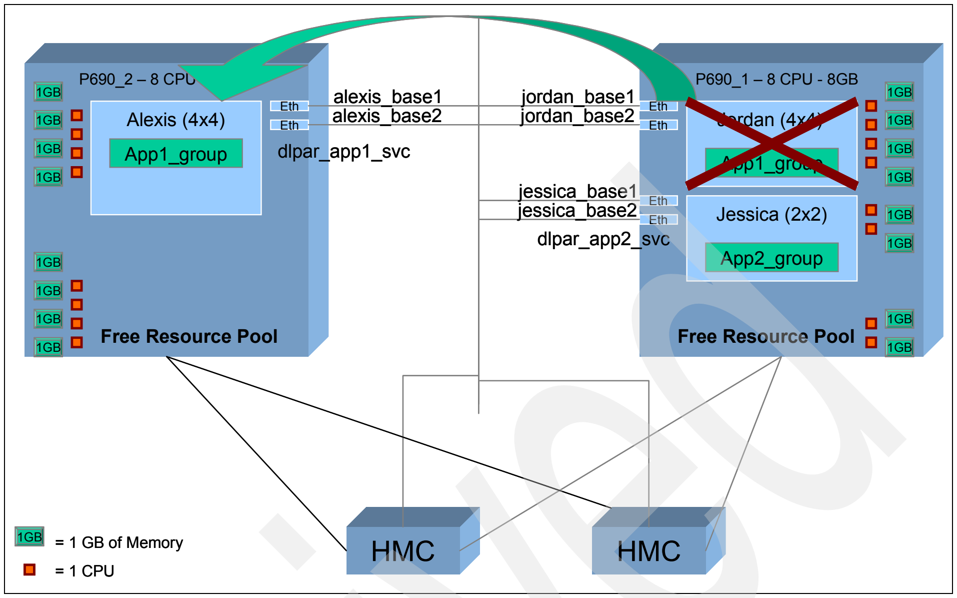


Figure 8 First production LPAR fallover

Node Alexis now has the same amount of resources as the original failing node.

In the second part of this scenario, we continue with the following setup:

- Jordan is offline.
- Jessica has two CPUs and 2 GB of memory.
- Alexis has four CPUs and 4 GB of memory.
- Free pool (frame 2) has four CPUs and 4 GB of memory.

We now cause node Jessica to fail by running **reboot -q**. Node Alexis takes over the app2 resource group and acquires the desired resources, as shown in Figure 9.

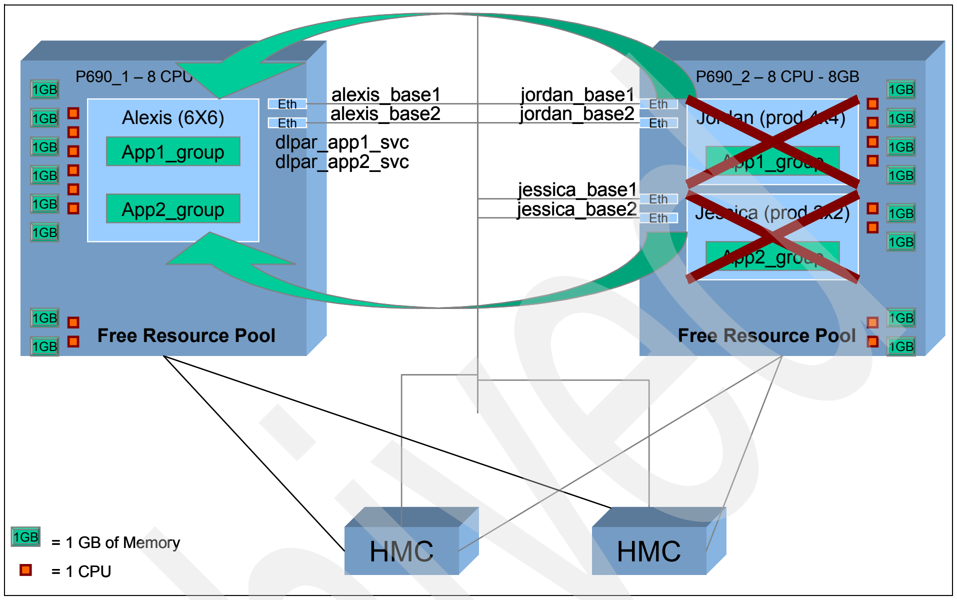


Figure 9   Second production LPAR fallover

Alexis ends up with its maximum partition setting of six CPUs and 6 GB of memory.

Scenario 4: Fallover production LPARs in reverse order

This is also a two part scenario. We start off exactly the same way we did with scenario 3.

We start off with cluster services running on all nodes. They currently have the following resources assigned to each node:

- Jordan (frame 1) has four CPUs and 4 GB of memory.
- Jessica (frame 1) has two CPUs and 4 GB of memory.
- Alexis (frame 2) has one CPU and 1 GB of memory.
- Free pool (frame 2) has seven CPUs and 7 GB of memory.

This time, we fail node Jessica first via our preferred method of **reboot -q**. This results in node Alexis acquiring the app2 resource group and the desired amount of resources, as shown in Figure 10 on page 36.

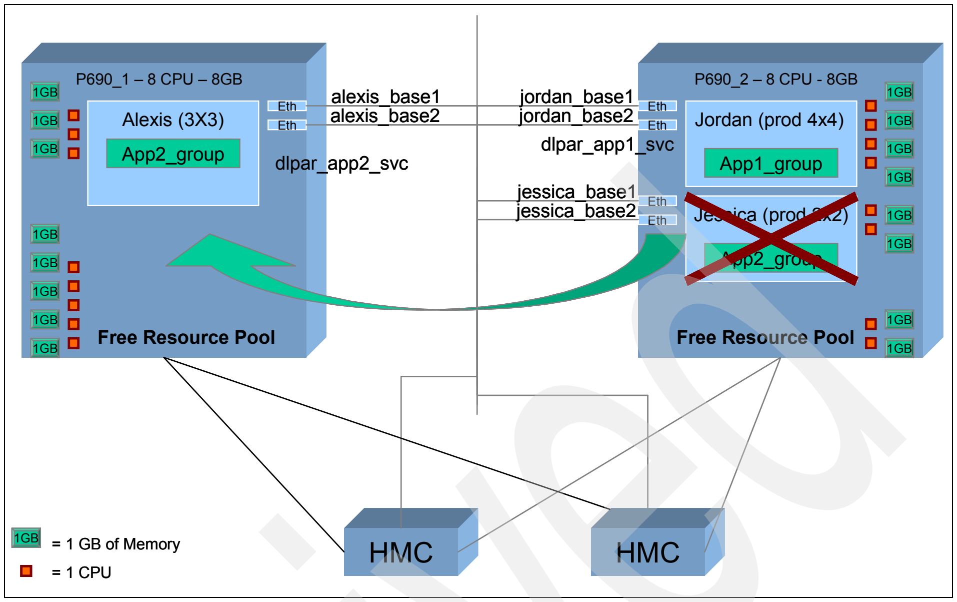


Figure 10   Fallover second production LPAR first

Alexis now has three CPUs and 3 GB of memory. Normally, the app2 resource group only has two CPUs and 2 GB of memory on node Jessica. Technically, this may be more resources than is necessary. This is a direct result of how application provisioning can result in a different amount of resources, depending on which LPAR/partition profile the resource group ends up on.

Let us continue the second part with the following setup:

- Jordan (frame 1) has four CPUs and 4 GB of memory.
- Jessica is offline.
- Alexis (frame 2) has three CPUs and 3 GB of memory.
- Free pool (frame 2) has five CPUs and 5 GB of memory.

We now fail node Jordan by running **`reboot -q`**. Node Alexis takes over the app1 resource group and acquires the desired resources, as shown in Figure 11. The end result is exactly the same as the previous scenario.

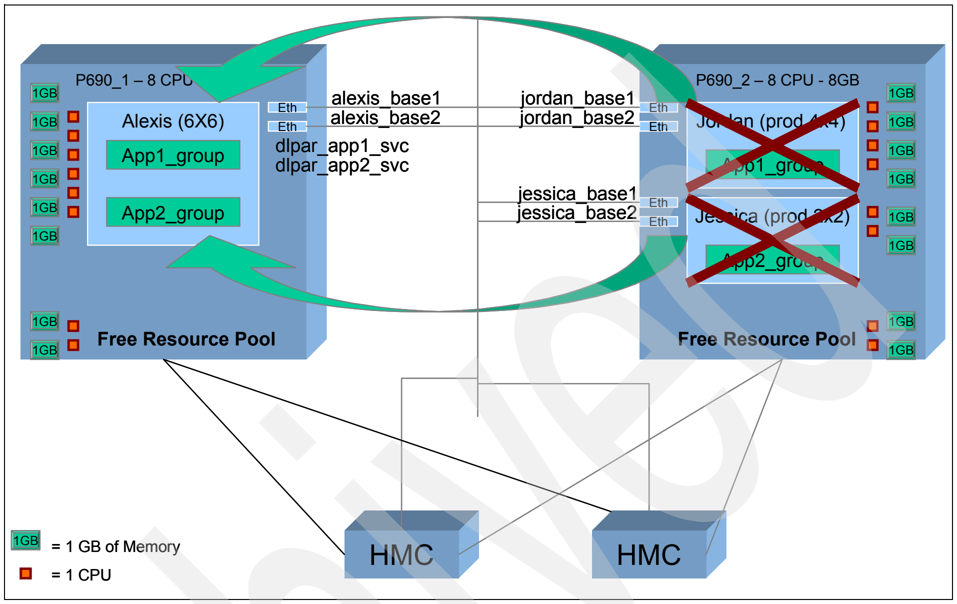


Figure 11   Results after second fallover

Scenario 5: Production re-acquisition via rg_move

In this scenario, we continue from where we left off in Scenario 4 after restarting nodes Jordan and Jessica in the cluster. We start with the following setup:

- Jordan (frame 1) has one CPU and 1 GB of memory.
- Jessica (frame 1) has one CPU and 1 GB of memory.
- Alexis (frame 2) has six CPUs and 6 GB of memory.
- Free pool (frame 1) has six CPUs and 6 GB of memory.

Node Alexis is currently hosting both resource groups. We run an rg_move of app2_rg from node Alexis back to its home node of Jessica. Alexis releases two CPUs and 2 GB of memory, while Jessica *only* acquires one CPU and 1 GB of memory, as shown in Figure 12 on page 38. This again is a direct result of the combination of application provisioning and the LPAR profile settings.

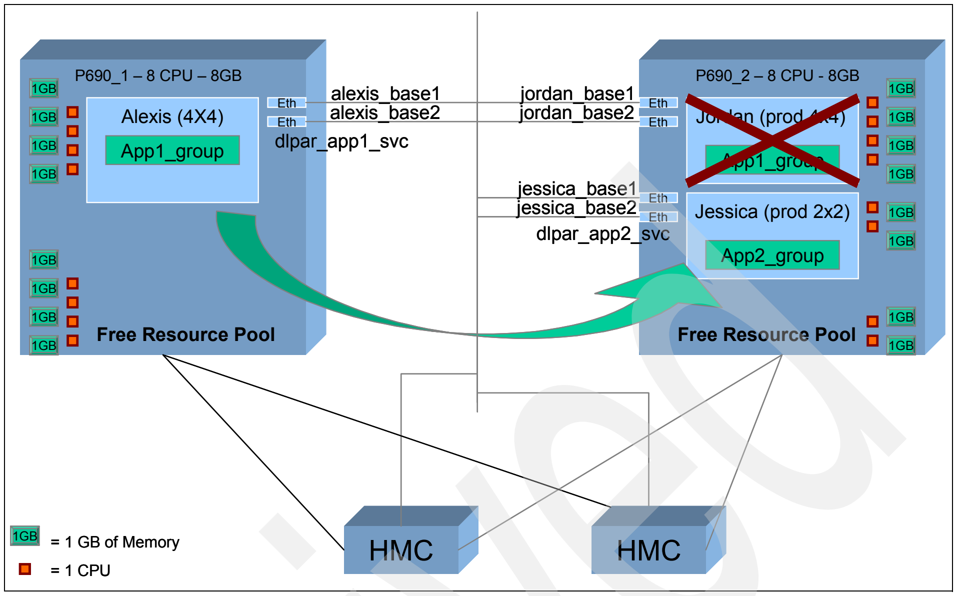


Figure 12   Resource group release and acquisition from rg_move

Scenario 6: Test HMC redundancy

In this scenario, we test the HMC redundancy by physically unplugging the network connection of one of the HMCs. We start off with cluster services running on all nodes and they currently have the following resources assigned to each of them:

- Jordan (frame 1) has four CPUs and 4 GB of memory.
- Jessica (frame 1) has two CPUs and 4 GB of memory.
- Alexis (frame 2) has one CPU and 1 GB of memory.
- Free pool (frame 2) has seven CPUs and 7 GB of memory.

We physically pulled the Ethernet cable from the HMC we have listed first in 192.168.100.69. We then failed node Jordan to initiate a fallover. We show this in Figure 13.

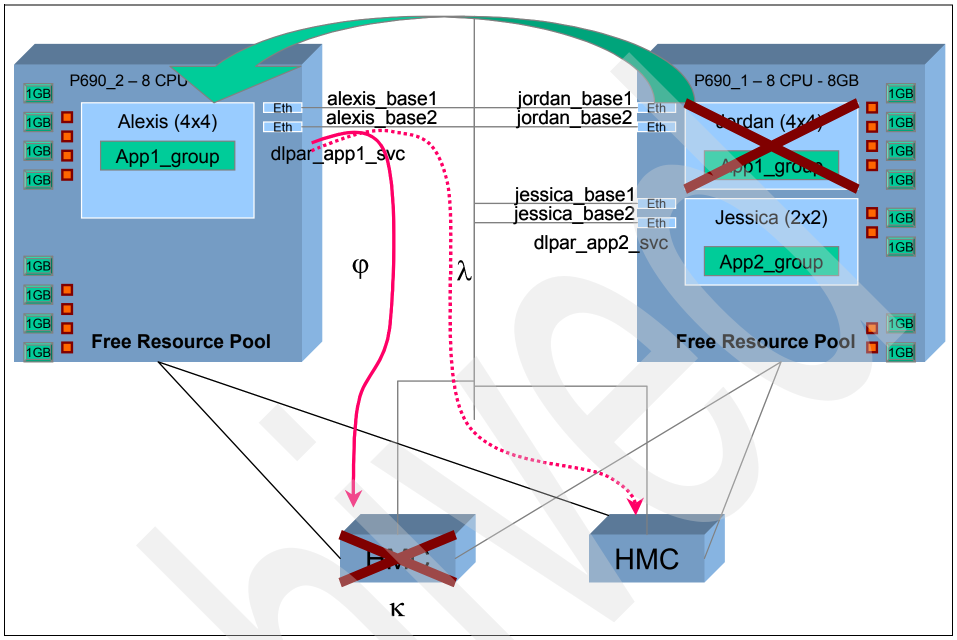


Figure 13 HMC redundancy test

During fallover and trying to access the HMC(s), the following occurs:

1. HACMP issues **ping** to the first HMC.
2. The first HMC is offline and does not respond.
3. HACMP issues **ping** to the second HMC, which is successful and continues to process dynamic LPAR command line operations.

The HMC test actions can be seen in /tmp/hacmp.out via the clhmcexec event utility.

HACMP and virtualization

During the writing of this redbook, the formal support announcement of HACMP and virtualization was released. While we do cover the support details in the following sections, the announcement can be found here:

```
http://w3-1.ibm.com/sales/systems/portal/_s.155/254?navID=f220s240&geoID=Al
l&prodID=IBM%20eServer%20And%20TotalStorage%20Products&docID=hacmpvio063005
```

Requirements

To use the integrated virtualization functions or CUoD of HACMP on POWER5, all LPAR nodes in the cluster should have at *least* the following levels installed:

- AIX 5L V5.3 Maintenance Level 5300-002 with APAR IY70082 and eFIX IY72974.
- HACMP
  - V5.1 with APAR IY66556 (or higher).
  - V5.2 with APAR IY68370 (or higher) and APAR IY68387.
  - V5.3.0.1with APAR IY73051 for ... support.
- RSCT
  - rsct.basic.hacmp.2.4.2.1
  - rsct.basic.rte.2.4.2.2
  - rsct.compat.basic.hacmp.2.4.2.0
- OpenSSH 3.4p1

  The OpenSSH software can be obtained from the AIX 5L V5.3 expansion pack or the Linux Toolbox CD. It can also be downloaded at:

  `http://sourceforge.net/projects/openssh-aix`

  OpenSSh for AIX has its own prerequisites (see paragraph "10.1.Requirements" in the README file that comes with the SSH package).
- Virtual I/O Server V1.1.2 with VIOS Fix Pack 6.2 and eFIX IY71303.062905.epkg.Z.
  - Fix Pack 6.2 can be downloaded from:

    `http://techsupport.services.ibm.com/server/vios/download/home.html`
  - eFIX IY72974 can be downloaded from:

    `ftp://ftp.software.ibm.com/.../efixes`
- OpenSSL SSL Cryptographic Libraries (OpenSSL), which can be downloaded from:

  `http://www-1.ibm.com/servers/aix/products/aixos/linux/download.html`

HMC attachment to the LPARs is required for proper management and dynamic LPAR capabilities. The HMC must also have at least the following levels installed:

- HMC V4R5 Build 20050519.1 or greater.

Important: APAR IY73051 for HACMP V5.3 is required to support micropartitioning, CUoD, and CBU functionality on POWER5 systems.

Application provisioning

All listed considerations in "Application provisioning" on page 32 are also available for the micropartitioning configuration. The granularity of what HACMP can manage is the physical CPU for dedicated processors, and virtual processors for shared processors configuration. Some scenarios of what you can do are shown below.

With micropartitioning, HACMP works with virtual processors, and not physical processors. In a mixed environment, HACMP verifies that ability to add the resources to respect the maximum value. For dedicated processor mode, the maximum value unit is the physical processor. For shared processor mode, the maximum value unit is the virtual processor (see Figure 14). The free pool resources is calculated by adding the desired capacity entitlement (CE) values of each partition in shared processor mode, and the physical processor in dedicated processor mode.

You cannot use capacity entitlement (tenth processor precision) as a value to define application provisioning in the HACMP menu.

HACMP does not verify if the partition is capped or uncapped; depending on the setup, you have to verify all cases.

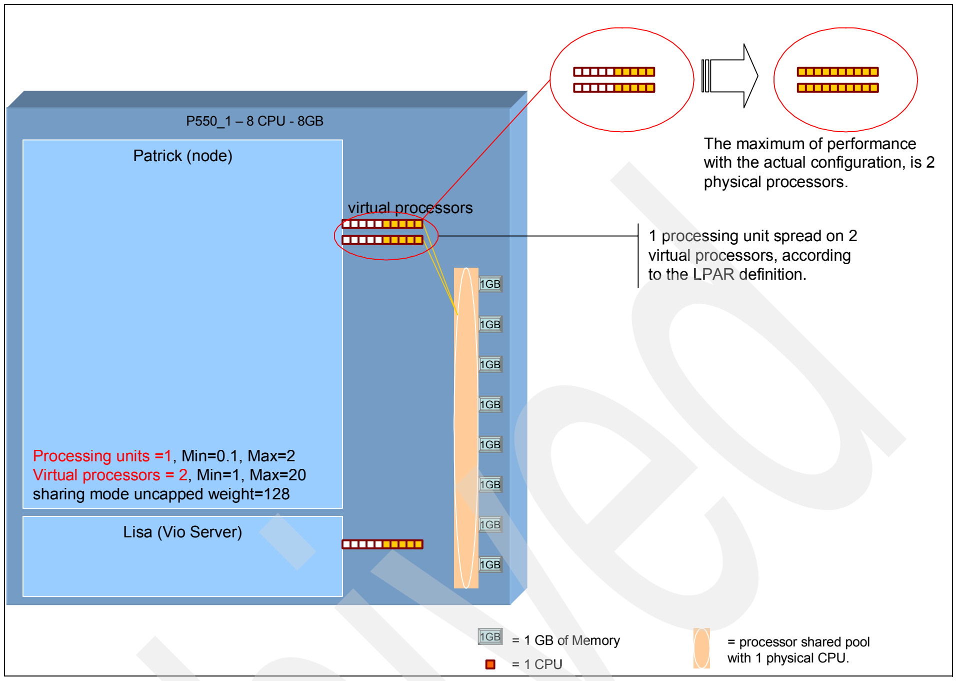

P550_1 – 8 CPU - 8GB
Patrick (node)
virtual processors
Processing units =1, Min=0.1, Max=2
Virtual processors = 2, Min=1, Max=20
sharing mode uncapped weight=128
Lisa (Vio Server)
1GB
The maximum of performance with the actual configuration, is 2 physical processors.
1 processing unit spread on 2 virtual processors, according to the LPAR definition.
= 1 GB of Memory
= 1 CPU
= processor shared pool with 1 physical CPU.

Figure 14 The attribution of capacity entitlement to a partition

The maximum value for one virtual processor is one physical processor (see Figure 14 on page 42). So the number of virtual processors attributed to a partition determines the maximum processing power that you can have in this partition.

Therefore, it is a good idea to use a virtual processor with HACMP instead of configuring the number of virtual processors in the partition for the worst case scenario (many applications falling over onto one node). In this case, the number of virtual processors is not optimized (more processing consumption is required for the Hypervisor).

If you are on a capped environment, you have to anticipate the number of virtual processors you can have in the partition to have the right performance for your application. Indeed, if you define a CE of 1 and you create four virtual processors, you have always one CE attributed, which is the equivalent of one physical processor, not four. In an uncapped environment, you do not have this problem (see Figure 15). The only limit on the partition is the number of virtual processors and the weight.

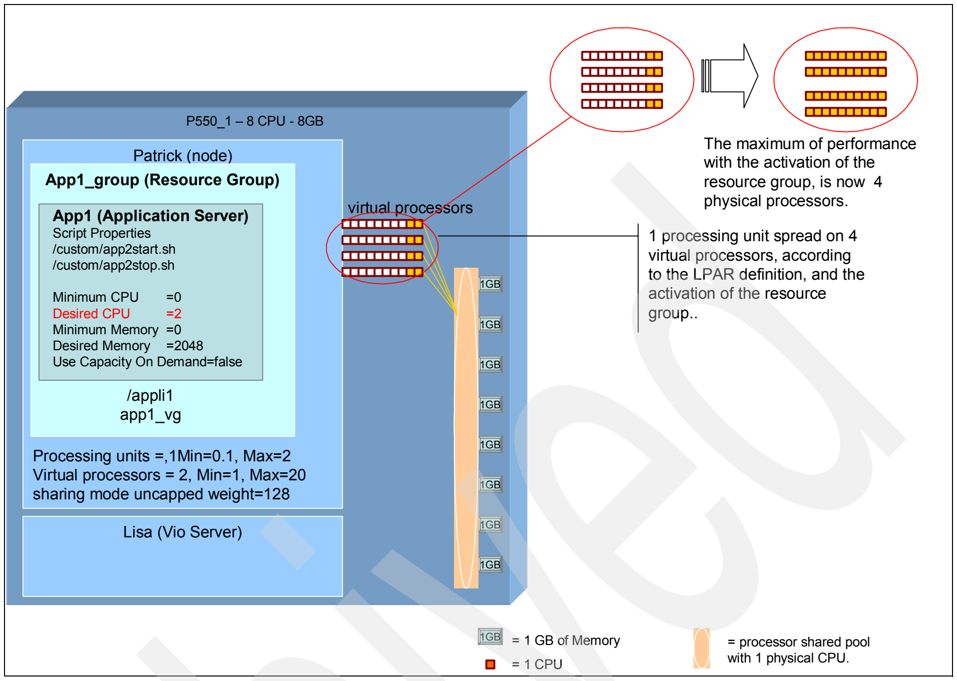

P550_1 – 8 CPU - 8GB
Patrick (node)
App1_group (Resource Group)
App1 (Application Server)
Script Properties
/custom/app2start.sh
/custom/app2stop.sh
Minimum CPU =0
Desired CPU =2
Minimum Memory =0
Desired Memory =2048
Use Capacity On Demand=false
/appli1
app1_vg
Processing units =,1Min=0.1, Max=2
Virtual processors = 2, Min=1, Max=20
sharing mode uncapped weight=128
Lisa (Vio Server)
virtual processors
1GB
The maximum of performance with the activation of the resource group, is now 4 physical processors.
1 processing unit spread on 4 virtual processors, according to the LPAR definition, and the activation of the resource group..
= 1 GB of Memory
= 1 CPU
= processor shared pool with 1 physical CPU.

Figure 15 Resource group activation

Let us discuss a micropartitioning example where we only use the processor's information. In this example, the minimum values for HACMP application provisioning is zero. We recommend this value to be zero to start the application server, even if we do not have available resources.

Table 7 presents the LPAR values as configured in the HMC, while Table 8 displays the real, active values.

Table 7 LPAR parameter values

LPAR name	LPAR values min/desired/max	Virtual processors min/desired/max	HACMP values	Processor.	Processor mode
patrick	0.1 / 1.0 / 2.0	1 / 2 / 20	app1 / 0 / 2		Shared
lisa	0.1 / 0.5 / 2.0	1 / 2 / 20	N/A		Shared
maelle	1.0 / 1.0 / 1.0	N/A	N/A	1	Dedicated
shawn	0.1 / 0.3 / 4.0	1 / 3 / 40	app2 / 0 / 1		Shared
lee	0.1 / 0.5 / 2.0	1 / 2 / 20			Shared
smokey	0.1 / 0.5 / 1.0	1 / 1 / 10			Shared

Table 8 Value used for calculation

LPAR name	LPAR values ①	Virtual processors min/desired/max	HACMP values	Processors
patrick	1	2	app1 / 0 / 2	
lisa	0.5	2	N/A	
maelle	1.0	N/A	N/A	1
shawn	0.3	3	app2 / 0 / 1	
lee	0.5			
smokey	0.5	1		

Note: These are sample calculations with the software version available at the time of the writing of this Redpaper. Check for PTF1 before you actually implement this type of configuration in production.

The free resources value is calculated by substracting 2.8 (the sum of column ① for one machine) from the total number of CPUs on the machine. The free resources on the patrick machine would therefore be 4 - 2.8 = 1.2, and the free resources on the shawn machine would be 4 - 1.3 = 2.7.

With one resource on node Patrick, the calculation would be 1 virtual processor (VP) (minimum LPAR) + 2 VP (as desired) = 3 active VPs when app2 is on.

The calculation to add one more on application app1 would be 1 VP (desired) + 3 VPs (current activity) - 1 VP (minimum for LPAR) - 2 VP = 1 VP to add. If this value is lower than the free resources (1.2), then HACMP will add the necessary resources.

Attention: If you want the application to start every time even if you do not have enough resources on the target machine, you have to put 0 as the minimum value on the resource (CPU and memory) in the menu of the application server dynamic LPAR definition.

This is applicable for dynamic LPAR with a dedicated mode processor.

The HACMP code permits you to adjust the LPAR memory block (LMB) size to the optimal size. The LMB size is obtained from the HMC, but the SMIT menu cannot increment beyond 256 MB.

The free resources are calculated by adding the number of dedicated processors and the capacity entitlement specified for each configured LPAR.

HACMP and virtualization configuration scenarios

Our first scenario is designed to assist you with the basic configuration of HACMP in a virtual environment. The second scenario is designed to bring out the functionalities of virtualization. These are complementary to the high availability mechanism that offers HACMP.

- Scenario 1 shows a configuration with two HACMP nodes in mutual takeover. This shows a basic configuration of HACMP.
- Scenario 2 shows a configuration with two HACMP clusters, including two nodes each.

Scenario 1: HACMP and virtual component outlook

HACMP could be used as usual in a virtualized environment. Of course, you have to respect the constraints mentioned in "Requirements" on page 40.

Figure 16 shows the components as seen by HACMP. For redundancy purposes, we define two Virtual I/O servers per machine. The AIX client's partition is connected to the VIOS partition through a virtual adapter. When you use the virtual SCSI to access the disks (vscsi0, vscsi1, and so on) multipath I/O (MPIO) is automatically used. That is why shared disks are available through two paths. The **`lspath`** command shows you the path managed by MPIO (Example 10 on page 47). There is no hardware errors reported to AIX by the virtual adapter, but HACMP uses RSCT to communicate the status of all communicating components.

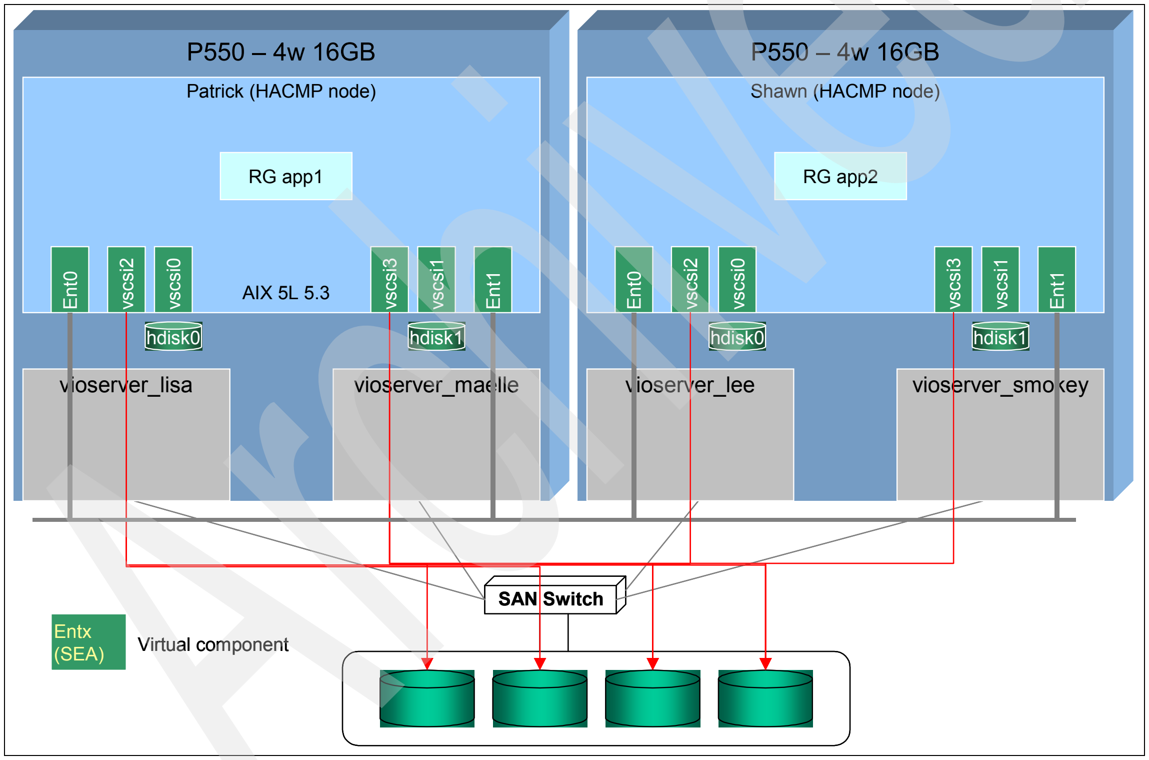


Figure 16 HACMP logical diagram outlook

Example 10   lspath command

```
patrick / > lspath
Enabled hdisk1 vscsi1
Enabled hdisk3 vscsi2
Enabled hdisk4 vscsi2
Enabled hdisk5 vscsi2
Enabled hdisk6 vscsi2
Enabled hdisk6 vscsi3
Enabled hdisk5 vscsi3
Enabled hdisk4 vscsi3
Enabled hdisk3 vscsi3
Enabled hdisk2 vscsi0
Enabled hdisk0 vscsi0
```

Example 11 shows the HACMP configuration used in our test cluster.

Example 11   Using cldump utility

```
patrick / > cldump
_____________________________________________________________________________
Cluster Name: app2_cluster
Cluster State: UP
Cluster Substate: UNSTABLE
_____________________________________________________________________________

Node Name: patrick              State: UP

  Network Name: net_diskhb_01      State: UP

    Address:                 Label: patrick2shawn      State: UP

  Network Name: net_ether_01       State: UP

    Address: 10.10.5.2       Label: patrick_base1      State: UP
    Address: 10.10.6.2       Label: patrick_base2      State: UP
    Address: 192.168.101.143 Label: vio_svc2           State: UP

Node Name: shawn                State: UP

  Network Name: net_diskhb_01      State: UP

    Address:                 Label: shawn2patrick      State: UP

  Network Name: net_ether_01       State: UP

    Address: 10.10.5.1       Label: shawn_base1        State: UP
    Address: 10.10.6.1       Label: shawn_base2        State: UP
    Address: 192.168.101.142 Label: vio_svc1           State: UP
```

```
Cluster Name: app2_cluster

Resource Group Name: app2_group
Startup Policy: Online On Home Node Only
Fallover Policy: Fallover To Next Priority Node In The List
Fallback Policy: Fallback To Higher Priority Node In The List
Site Policy: ignore
Priority Override Information:
    Primary Instance POL:
Node                         Group State
---------------------------- ---------------
shawn                        ONLINE
patrick                      OFFLINE

Resource Group Name: app1_group
Startup Policy: Online On Home Node Only
Fallover Policy: Fallover To Next Priority Node In The List
Fallback Policy: Fallback To Higher Priority Node In The List
Site Policy: ignore
Priority Override Information:
    Primary Instance POL:
Node                         Group State
---------------------------- ---------------
patrick                      ONLINE
shawn                        OFFLINE
```

The virtual I/O (VIO) server has to be configured to offer the virtualized components to the partition.

Figure 17 on page 49 shows the details of the hardware components that were used. The legend distinguishes the virtual components from the physical components. Note that in our test, AIX and HACMP are based only on virtual components. We use four virtual SCSI on each node, two of which are used for the mirrored system disk, and two are used for the shared disks. All access is spread across two VIO servers.

The shared disks for both client partitions have to be defined as hdisk on the target definition in the VIO server.

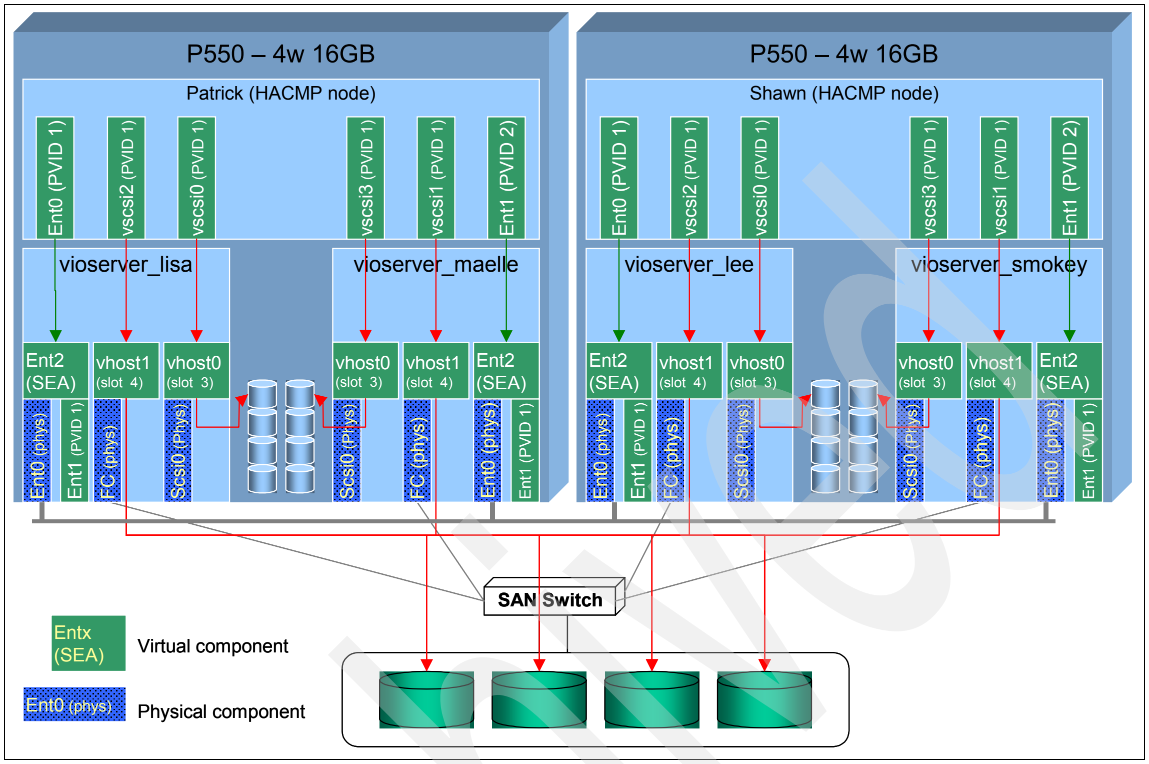
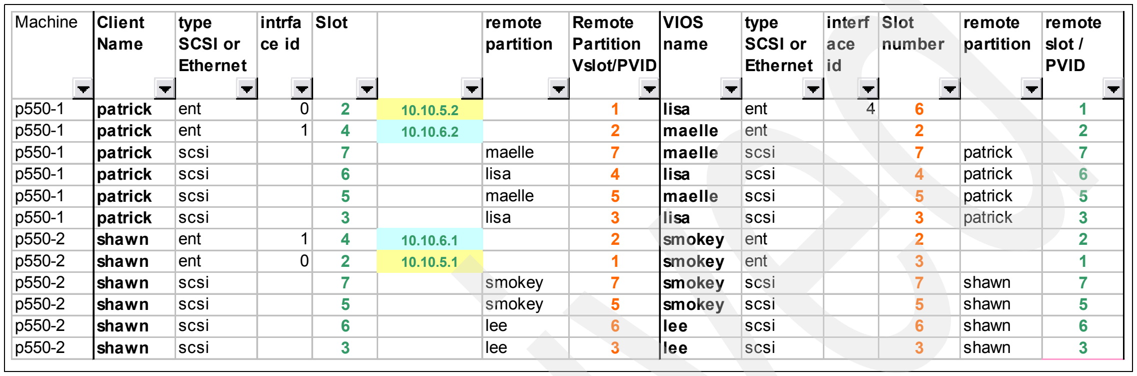


Figure 17   Virtualization architecture diagram

Virtualization configuration

Before you plan your configuration, we suggest you refer to *Advanced POWER Virtualization on IBM @server p5 Servers: Introduction and Basic Configuration*, SG24-7940. We do the following steps to implement our virtualization configuration:

1. Plan the configuration. Figure 18 gives you an overview of how we planned our configuration.



Machine	Client Name	type SCSI or Ethernet	intrface id	Slot		remote partition	Remote Partition Vslot/PVID	VIOS name	type SCSI or Ethernet	interface id	Slot number	remote partition	remote slot / PVID
p550-1	**patrick**	ent	0	2	10.10.5.2		1	**lisa**	ent	4	6		1
p550-1	**patrick**	ent	1	4	10.10.6.2		2	**maelle**	ent		2		2
p550-1	**patrick**	scsi		7		maelle	7	**maelle**	scsi		7	patrick	7
p550-1	**patrick**	scsi		6		lisa	4	**lisa**	scsi		4	patrick	6
p550-1	**patrick**	scsi		5		maelle	5	**maelle**	scsi		5	patrick	5
p550-1	**patrick**	scsi		3		lisa	3	**lisa**	scsi		3	patrick	3
p550-2	**shawn**	ent	1	4	10.10.6.1		2	**smokey**	ent		2		2
p550-2	**shawn**	ent	0	2	10.10.5.1		1	**smokey**	ent		3		1
p550-2	**shawn**	scsi		7		smokey	7	**smokey**	scsi		7	shawn	7
p550-2	**shawn**	scsi		5		smokey	5	**smokey**	scsi		5	shawn	5
p550-2	**shawn**	scsi		6		lee	6	**lee**	scsi		6	shawn	6
p550-2	**shawn**	scsi		3		lee	3	**lee**	scsi		3	shawn	3

Figure 18 Virtualization configuration

2. Create the LPAR configuration on the HMC. Each VIOS partition is defined with one virtual Ethernet and two virtual SCSI adapters. The virtual Ethernet adapter is defined as a trunk adapter (ent1). Both virtual scsi adapters are defined as servers (vhost0 and vhost1).
3. Install the VIOS partition and then update it with the **`updateios`** command.
4. Set up the shared Ethernet adapter (SEA), as shown in Example 12.

Example 12 Creating the shared Ethernet adapter

```
mkvdev -sea ent0 -vadapter ent1 default ent1 -defaultid 1

mktcpip -hostname vioserver_lisa -inetaddr 192.168.100.220 -netmask
255.255.255.0 -gateway 192.168.100.60 -start
```

5. Create the virtual disks. In our cluster, we create two types of disks for the client. On node shawn, we create both target disks on the VIOS partition as logical volumes (see Example 13 on page 39). On node patrick, we create both target disks on VIOS partition as hdisks (see Example 13 on page 51).

Example 13 Creating the virtual disks devices

```
mkvg -f -vg shawnvg hdisk1
mklv -lv shawn_lv shawnvg 30G
mkvdev -vdev shawn_lv -vadapter vhost0 -dev vshawn_disk
mkvdev -vdev hdisk1 -vadapter vhost0 -dev vpatrick_disk
```

6. Install the system and HACMP in the client partition.
7. Mirror the system disks and change the bootlist.
8. If you want to use the dynamic LPAR or CUoD function, install and configure SSH, as described in "Install and configure SSH on HACMP nodes" on page 17.

Test results

We performed the following tests and obtained the following test results.

rootvg mirroring

By shutting down VIOS, the access on one part of the disk becomes unavailable. The mirror allows you to continue working. At the level of AIX that we are using, to make the disks available again, you have to extract the disk from the rootvg definition and then add the same disk and do the mirroring. Then you have to run the **bosboot** command.

adapter swap

HACMP uses RSCT to track the state of the communication interfaces or devices. Figure 19 shows the identification of a failure on a virtual adapter. In this test, we disconnect an Ethernet cable to simulate a physical network failure.

```
07/05 18:32:45.542: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:33:05.542: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:33:25.548: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:33:40.789: Heartbeat was NOT received. Missed HBs: 1. Limit: 10
07/05 18:33:42.789: Heartbeat was NOT received. Missed HBs: 2. Limit: 10
07/05 18:33:42.789: Starting sending ICMP ECHOs.
07/05 18:33:42.789: Invoking netmon to find status of local adapter.
07/05 18:33:44.789: Heartbeat was NOT received. Missed HBs: 3. Limit: 10
07/05 18:33:46.789: Heartbeat was NOT received. Missed HBs: 4. Limit: 10
07/05 18:33:47.497: netmon response: Adapter seems down
...
07/05 18:33:58.809: Heartbeat was NOT received. Missed HBs: 10. Limit: 10
07/05 18:33:58.809: Local adapter is down: issuing notification for local adapter
07/05 18:33:58.809: Adapter status successfully sent.
07/05 18:33:58.945: Dispatching netmon request while another in progress.
07/05 18:33:58.945: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:33:59.497: netmon response: Adapter seems down
07/05 18:33:59.497: Adapter status successfully sent.
07/05 18:41:42.982: netmon response: Adapter is up
07/05 18:41:42.982: Bind to broadcast address succeeded.
07/05 18:41:42.982: Adapter status successfully sent.
07/05 18:41:45.849: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.859: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.859: Received a STOP MONITOR command.
07/05 18:41:46.861: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.861: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.861: Received a START HB command. Destination: 10.10.5.2.
07/05 18:41:46.861: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.861: Received a START MONITOR command.
07/05 18:41:46.861: Address: 10.10.5.2 How often: 2000 msec Sensitivity: 10 Configuration Instance: 11
07/05 18:41:46.862: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:46.862: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:47.660: netmon response: Adapter is up
07/05 18:41:51.871: Received a SEND MSG command. Dst: 10.10.5.2.
07/05 18:41:57.871: Received a SEND MSG command. Dst: 10.10.5.2.
```

Figure 19   nim.topsvcs.en2.app2_cluster

In this example, before 18:33, the network connection was available. At 18:33:30, we disconnect the Ethernet cable. The topology services identifies, by missed heartbeat, that the adapter en2 is down at 18:33:59. At 18:34:02, HACMP runs **`swap_adapter`**.

A VIO server down.

In this test, vioserver_lisa is down. All operations continue to work through the second path. MPIO does its work by preserving the access to the data through the second VIO server (vioserver_maelle). Of course, rootvg is always active on the mirrored disk (Figure 20). The **`lspath`** command shows that one path has failed in regard to shared disk access.

As described in "rootvg mirroring" on page 51, when the VIO server is up, you have to perform the operation manually. The **`lspath`** command shows you the status of each path (see Figure 14 on page 54).You can force the attempt with the **`chpath`** command or **`smit mpiopath_enable_all`** command. Figure 20 shows the results of running this command. The virtual Ethernet adapter is joined automatically by the join_interface event.

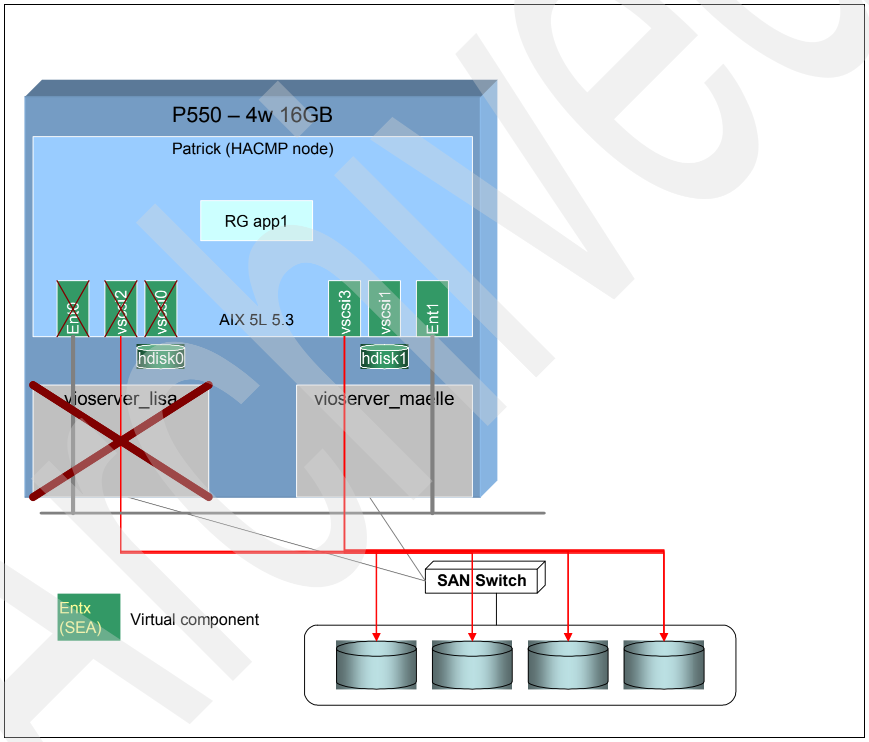


Figure 20 VIO server failure

Example 14 displays the missing path information for the virtual disks.

Example 14 lspath command with path missing

```
patrick / > lspath
Enabled hdisk1 vscsi1
Missing hdisk3 vscsi2
Missing hdisk4 vscsi2
Missing hdisk5 vscsi2
Missing hdisk6 vscsi2
Failed  hdisk6 vscsi3
Failed  hdisk5 vscsi3
Failed  hdisk4 vscsi3
Failed  hdisk3 vscsi3
Enabled hdisk2 vscsi0
Missing hdisk0 vscsi0
```

When one VIO server is down, the **lspath** command produces output that is similar to the output shown in Example 15.

Example 15 Example of lspath command output when one VIO server is down

```
patrick / > lspath
Enabled hdisk1 vscsi1
Missing hdisk3 vscsi2
Missing hdisk4 vscsi2
Missing hdisk5 vscsi2
Missing hdisk6 vscsi2
Enabled hdisk6 vscsi3
Enabled hdisk5 vscsi3
Enabled hdisk4 vscsi3
Enabled hdisk3 vscsi3
Enabled hdisk2 vscsi0
Missing hdisk0 vscsi0
```

One node fallover

The final HACMP behavior is the same for these tests:

- Two VIO servers are down.
- Each Ethernet adapter is disconnected from both VIO servers.
- One node is halted.

HACMP has finished its logical processing.

Scenario 2: Performance and architecture considerations

We try to go further and then add another cluster on the same infrastructure. This scenario includes two HACMP clusters. We also define two virtual SCSI servers, as shown in Figure 21.

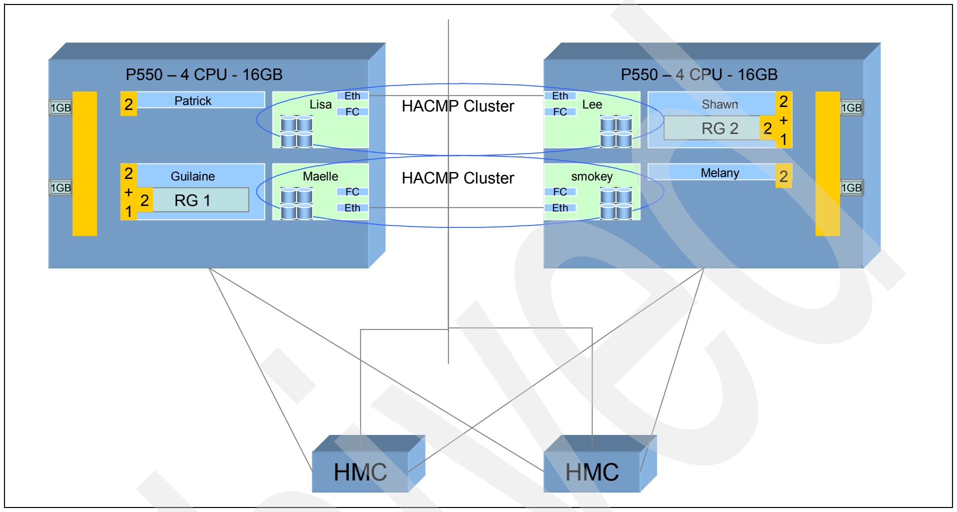


Figure 21   Two clusters with two CECs and two HMCs

The benefit of this architecture is that you manage two simple clusters with two machines, which has the following advantages:

- With this setup, you have a standby node, which reduces the risk of downtime.
- Each application has its own cluster, and the ease of administration will help you better manage your environment.
- By configuring the virtualization, you can manage the application provisioning.
- Using the HACMP configuration assistant will ensure a quick cluster configuration.

The RG1 resource group needs more CPU resources. By configuring the uncapped weight value on each partition, you can support an environment where the two resource groups are comparable. HACMP activates the resources (the number of virtual processors) associated with the application server. The calculation mechanism is the same one in “Application provisioning” on page 7.

In this configuration, we use two functions: one from HACMP with the application provisioning feature and the second one with the micropartitioning feature that comes with Advance Power Virtualization (APV). Instead of stopping another partition to free some resources, we use the uncapped weight feature to give priority to the most important partition.

The micropartitioning parameters (shown in Table 9) are independent of the HACMP parameters.

Table 9 Parameters list of cluster configuration

LPAR name	**CE min/desired/max**	**HACMP dynamic LPAR min/desired**	**Uncapped weight**	**Virtual Processors min/desired/ax**
Patrick	0.5 / 2.0 / 4.0	app2 0 / 2	200	1 / 2 / 40
Guilaine	0.5 / 2.0 / 4.0	app1 0 / 2	10	1 / 2 / 40
Maelle	0.1 / 0.5 / 1	N/A	128	1 / 2 / 10
Lisa	0.1 / 0.5 / 1	N/A	128	1 / 2 / 10

Try the following test. We fall over one node to the other one, as shown in Figure 22 on page 57. At the same time, we collect performance statistics on each partition. Since LPAR patrick has an uncapped weight value of 200 and LPAR guilaine has one of 10, give more priority to patrick.

The uncapped weight is a number between zero through 255 that you set for each uncapped partition in the shared processor pool. 255 is the highest weight. Any available unused capacity is distributed to the partitions in proportion to the other uncapped partitions.

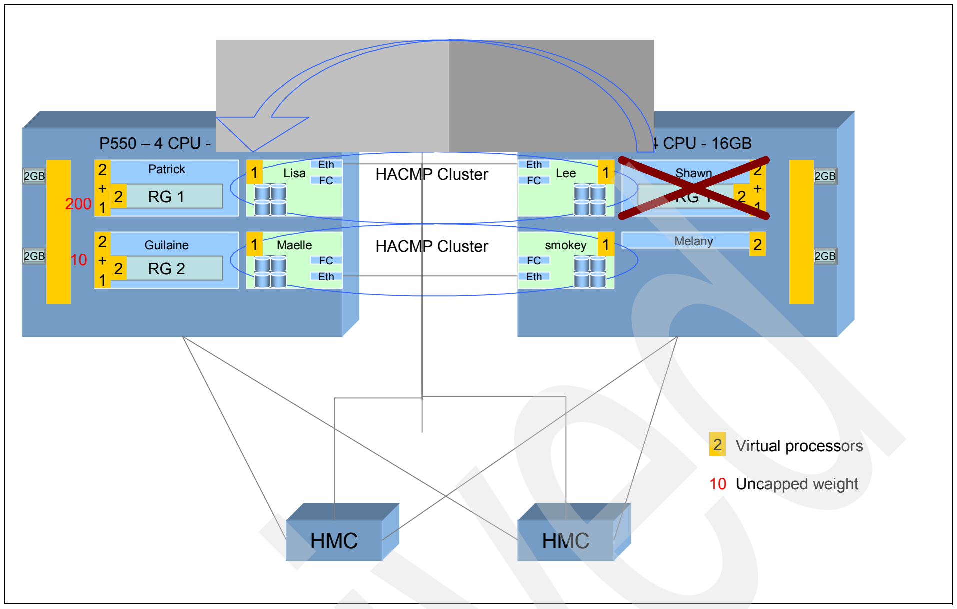
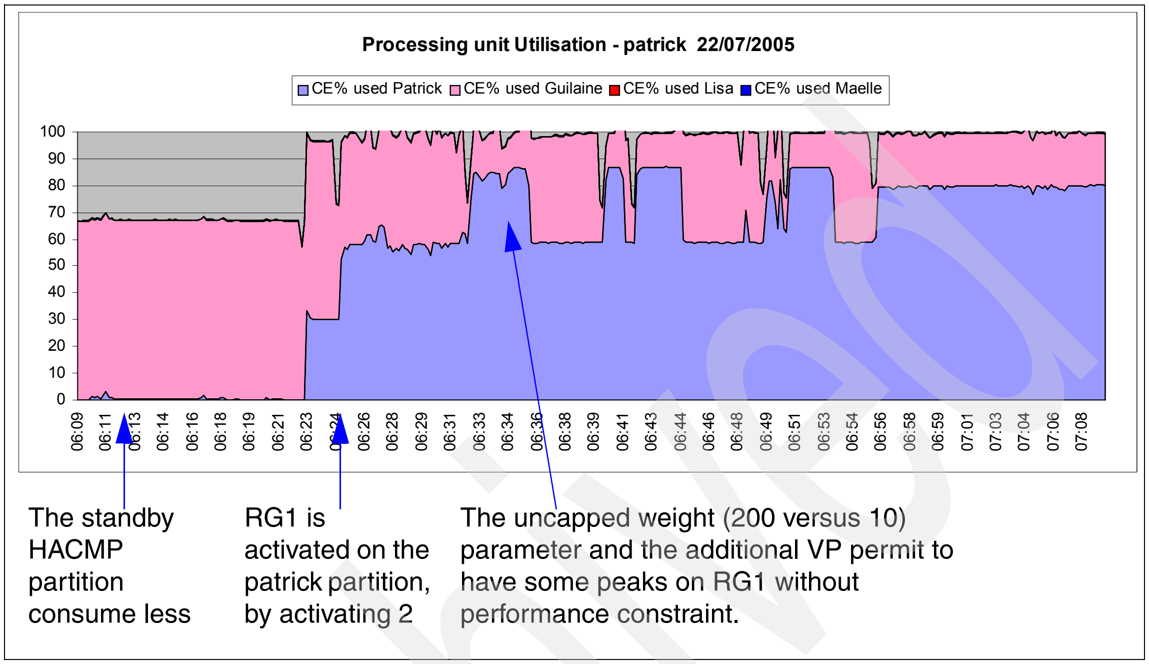


Figure 22 Takeover on one machine with privileged processing of RG1

The graph in Figure 23 represents the processing units activated on each partition on one machine of the configuration.



The standby HACMP partition consume less

RG1 is activated on the patrick partition, by activating 2

The uncapped weight (200 versus 10) parameter and the additional VP permit to have some peaks on RG1 without performance constraint.

Figure 23 Graph of the effective processing unit activated in each partition

Notices

This document created or updated on February 19, 2006.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  ibm.com/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 905, 11501 Burnet Road
  Austin, Texas 78758-3493 U.S.A.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX 5L™	IBM®	pSeries®
AIX®	MVS™	Redbooks™
@server®	POWER™	Redbooks (logo) ™
@server®	POWER4™	Tivoli®
HACMP™	POWER5™	

The following terms are trademarks of other companies:

Solaris, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.