Installing and Managing HP-UX Virtual Partitions (vPars)

Eighth Edition

Manufacturing Part Number: T1335-90051
March 2006

United States

Legal Notices

Warranty

Restricted Rights Legend

Copyright Notices

Publication History

The manual publication date and part number indicate its current edition. The publication date will change when a new edition is released. The manual part number will change when extensive changes are made.

To ensure that you receive the new editions, you should subscribe to the appropriate product support service. See your HP sales representative for details.

- Eighth Edition: March 2006 T1335-90051
  (includes vPars version A.04.02 on HP-UX 11i v2)
- Seventh Edition: October 2005 and January 2006 Update, T1335-90041
  (includes vPars version A.03.03 on HP-UX 11i v1)
- Sixth Edition: July 2005 and September 2005 Update, T1335-90038
  (includes vPars version A.04.01 on HP-UX 11i v2)
- Fifth Edition: December 2004, T1335-90031
  (includes vPars version A.03.02 on HP-UX 11i v1)
- Fourth Edition: April 2004, T1335-90027
  (includes vPars version A.03.01 on HP-UX 11i v1)
- Third Edition: November 2002, T1335-90018
  (includes vPars version A.02.02 and A.02.03 on HP-UX 11i v1)
- Second Edition: June 2002, T1335-90012
  (includes vPars version A.02.01 on HP-UX 11i v1)
- First Edition: November 2001, T1335-90001
  (includes vPars version A.01.01 on HP-UX 11i v1)

IMPORTANT New information may have been developed after the time of this printing. For the most current information, check the Hewlett-Packard documentation web site at the following URL:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

Contents

Tables

1 Introduction

This chapter covers:

- Information on This Document
- What Is vPars?
- Why Use vPars?
- Supported Environments
- Product Interaction
- Ordering vPars

Information on This Document

Intended Audience

This document is written for system administrators to help them learn and manage the product HP-UX Virtual Partitions (vPars).

How This Book is Organized

Chapter 1 covers a brief introduction to vPars and its interaction with other HP-UX products.

Chapter 2 covers conceptual material about vPars and its components.

Chapter 3 covers planning your vPars environment.

Chapter 4 covers vPars installation.

Chapter 5 covers vPars commands and managing vPars.

Chapter 6 covers managing the hardware resources for virtual partitions.

Chapter 7 covers vPars crash processing and recovery.

Chapter 8 covers the vPars GUI vparmgr.

Where to Get the Latest Version of This Document

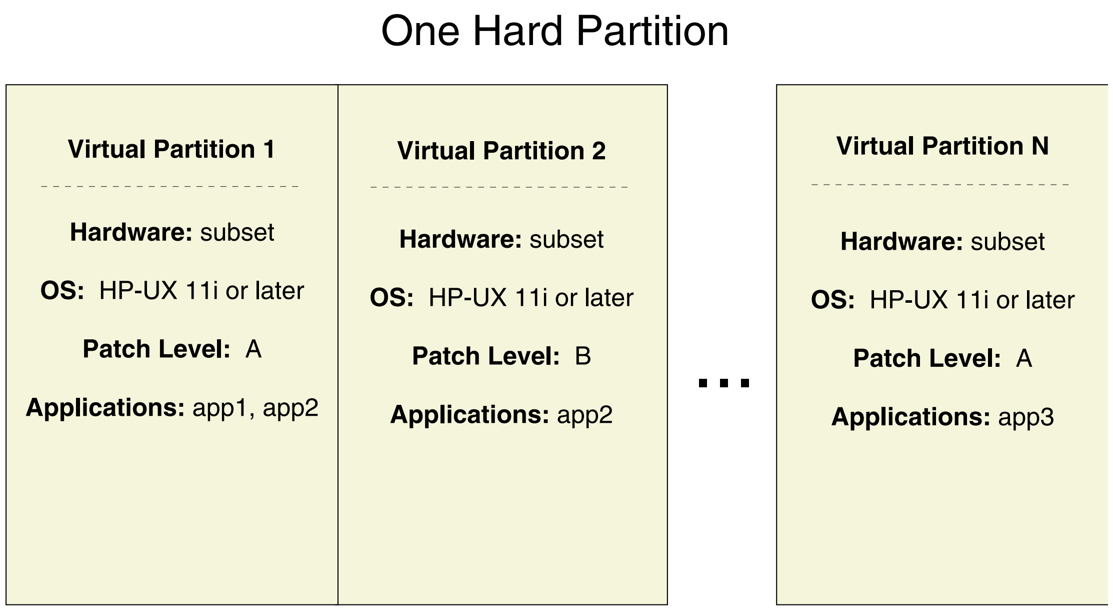
Go to the HP Documentation web site at

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

What Is vPars?

The vPars (Virtual Partitions) product allows you to run multiple instances of HP-UX simultaneously on one hard partition by dividing the hard partition further into **virtual partitions**. Each virtual partition is assigned its own subset of hardware, runs a separate instance of HP-UX, and hosts its own set of applications. Because each HP-UX instance is isolated from all other instances, vPars provides application and OS (Operating System) fault isolation. Each instance of HP-UX can have different patches and have a different kernel.

Figure 1-1 vPars Conceptual Diagram



NOTE **Definitions**

In this document, we have redefined the terms virtual partitions, nPartitions, and hard partitions:

A **complex** is the entire partitionable server, including both cabinets, all cells, IO chassis, cables, and power and utility components.

A **cabinet** is the Superdome’s hardware "box", which contains the cells, Guardian Service Processor (GSP), internal IO chassis, IO fans, cabinet fans, and power supplies. A complex has up to two cabinets.

Figure 1-2A Superdome Cabinet

A **hard partition** is any isolated hardware environment, such as an nPartition within a Superdome complex or an entire rp7400/N4000 server.

A **nPartition** is a subset of a complex that divides the complex into groups of cell boards where each group operates independently of other groups. an nPartition can run a single instance of HP-UX or be further divided into virtual partitions.

A **virtual partition** is a software partition of a hard partition where each virtual partition contains an instance of HP-UX. Though a hard partition can contain multiple virtual partitions, the inverse is not true. A virtual partition cannot span a hard partition boundary.

Product Features

- A single hard partition can be divided into multiple virtual partitions.
- Each virtual partition runs its own instance of HP-UX. Thus, different applications or multiple instances of the same application can run in different virtual partitions on the same hard partition at the same time without conflicts.
- Each virtual partition is assigned its own resources (CPU, memory, and IO), so there are no resource conflicts between virtual partitions.
- Virtual partitions can be of different patch levels.
- Virtual partitions can be individually reconfigured and rebooted (for patches and other changes that require a reboot).
- Users on one virtual partition cannot access files or file systems on other partitions (unless the file systems are NFS-mounted or access is otherwise given through networking or for cluster-aware volume groups used within ServiceGuard). Further, users configured on one virtual partition does not imply a presence on any other partition.
- Software-related kernel panics[1], resource exhaustion failures, and subsequent reboots in one virtual partition do not affect any other virtual partition.
- CPUs available at boot time can be added to or removed from a virtual partition without rebooting.

1. Except if the vPars software product itself panics.

Why Use vPars?

The following explains some of the advantages of using vPars:

vPars Increases Server Utilization and Isolates OS and Application Faults

In certain environments one entire server is dedicated to a single application. When the demand for that application is not at peak, such as during non-business hours, the server is underutilized. If many servers are configured this way, you have many servers that are being underutilized. You can minimize investment and operational costs by consolidating servers and running multiple applications on one server; however, this leaves all applications vulnerable to problems if any one application or their now single OS has problems.

vPars provides a software-based solution that supports isolating OSs and their applications within virtual partitions; thus, OS or application problems in one virtual partition do not affect OSs or applications running in other partitions.

vPars also allows consolidation of underutilized servers into one faster server where applications may not be permitted to affect one another, such as in the case of an ISP running many small e-services application servers.

vPars Provides Flexibility Through Multiple but Independent OS Instances

vPars offers flexibility by allowing different HP-UX instances and patch levels to run on the same server.

vPars Provides Flexibility Through Dynamic CPU Migration

vPars allows you to reassign CPUs from one virtual partition to another without rebooting.

Two virtual partitions that have different CPU utilization peak times can have processors moved between them. For example, a transaction server used primarily during business hours could have CPUs reassigned overnight to a report server. Such reassignments can be automated, for example, via cron.

Because vPars assigns specific hardware resources to specific virtual partitions, a user on the transaction server at night is not affected by the reports server’s huge consumption of processing power. A virtual partition uses only the CPUs that you assign to it; CPUs are not time-sliced across the virtual partitions.

Supported Environments

Hardware and Firmware

This information on supported hardware and required firmware versions has been moved to the document *HP-UX Virtual Partitions Ordering and Configuration Guide*, available at:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

NOTE Updating Firmware on the rp5470/L3000 and rp7400/N4000

Installing firmware patches on these servers requires additional steps in a vPars environment. Please see “Notes on Installing Server Firmware” on page 68.

Operating Systems

vPars versions A.01.xx-A.03.xx:

- All virtual partitions must run HP-UX 11i v1 (11.11) using the December 2000 Release or later in 64-bit mode on PA-RISC platforms. vPars A.03.xx and earlier do not support HP-UX 11i v2; you must use vPars A.04.

vPars versions A.04.xx:

- All virtual partitions must run HP-UX 11i *v2* (11.23) May 2005 Release or later in 64-bit mode on PA-RISC or Integrity platforms.

CAUTION All OS and vPars versions within a given vPars environment must be the same. Having a mix of OS versions (for example, 11.11 and 11.23) or a mix of vPars versions (for example, A.03.03 and A.04.01) is not allowed. If one virtual partition is running HP-UX 11.23, all the virtual partitions within the same vPars environment must be running 11.23. Or if one virtual partition is running A.04.01, all the virtual partitions within the same vPars environment must be running A.04.01.

HP Product Interaction

- **nPartitions**

  (vPars A.03.xx and earlier) Parmgr Requirements: For information on which version of Partition Manager (parmgr) is required with vPars, please see the document *Read Before Installing HP-UX Virtual Partitions* available at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions. For information on installing Partition Manager, see "Installing and Removing vPars-related Bundles" on page 62.

  vPars on nPartitions Conceptual Points:

  — Only one vPars Monitor is booted per nPartition.

  — Virtual partitions exist within an nPartition, but they cannot span across nPartitions.

  — Each virtual partition within a given nPartition can be assigned a subset of only the hardware assigned to the nPartition. Furthermore, only the *active* hardware assigned to the nPartition can be used by the virtual partitions within the nPartition.

  — nPartitions remain isolated from other nPartitions, regardless of whether vPars is installed. You can have virtual partitions installed within an nPartition without affecting the other nPartitions.

  — The vPars database is entirely separate from the **nPartition complex profile data**. Therefore, a change in the vPars partition database does not change any complex profile data. For an example on changing information in both the vPars partition database and the nPartition complex profile, see "Using Primary and Alternate Paths with nPartitions" on page 140.

  — If there is a **pending reboot for reconfiguration** (RFR) for the involved nPartition, no virtual partitions will be rebooted until all the virtual partitions within the given nPartition are shut down and the involved vPars Monitor is rebooted. This implies that the target virtual partition of the `vparload`, `vparboot`, and `vparreset` commands will not boot until all virtual partitions within the nPartition have been shut down and the vPars Monitor is rebooted.

  — For more information on performing nPartition operations within a virtual partition, see "Managing: Performing nPartition Operations" on page 129. For more information on using the `-R` and `-r` options of the `shutdown` and `reboot` commands used in a RFR, see "shutdown and reboot commands" on page 26.

  — For more information, see "Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)" on page 135 and the vPars Monitor command "reboot [mode]" on page 107.

  — For a given nPartition, the **Virtual Front Panel** (VFP) displays an OS heartbeat whenever at least one virtual partition within the nPartition is up.

  — All vPars within an nPartition share the same **console device**. For a given nPartition, this is the nPartition's console. For more information on the console and console logs, see "Virtual Consoles" on page 39 and "nPartition Logs" on page 41.

  For more information on the nPartition hardware, see the manual *HP Systems Partition Guide*.

NOTE Note: unlike the rp7400/N4000, on a Superdome and other nPartition servers, the first element of the hardware path of the `ioscan` output is the cell number.

For example, on the rp7400/N4000 the `ioscan` output shows:

```
0/0       ba              Local PCI Bus Adapter (782)
```

However, on a Superdome, the first element of the hardware path is the cell number. So, if the cell number is `4`, the `ioscan` output shows:

```
4/0/0     ba               Local PCI Bus Adapter (782)
```

- **PCI OL* (On-Line Addition and Replacement)**

  Except for the function stated below, OL* for PCI slots works the same on a vPars server as it does on a non-vPars server. Note that you can execute PCI OL* functions only on the PCI slots that the virtual partition owns.

  PCI doorbells (the physical attention button on the system) are supported beginning with the HP-UX December 20003 HWE release and vPars A.03.01.

  (PA-RISC only) In a vPars system, a reboot of the virtual partition does not power on a slot that was powered off prior to the reboot. If you wish to power on the slot, you need to power on the slot manually after the reboot using the `rad` command: `rad -i` *`slot_id`*.

  For changes in the use of OLAR on HP-UX 11i v2 (11.23), please see the 11.23 OLAR documentation at http://docs.hp.com.

- **iCAP (Instant Capacity, formerly known as iCOD (Instant Capacity on Demand)) and PPU (Pay Per Use)**

  iCAP allows online activation of iCAP CPUs; PPU allows purchasing CPU time of processors.

  iCAP or PPU and vPars version information:

  — Beginning with vPars version A.04.01, vPars now supports PPU Active CPU as well as PPU Percent Utilization)

  — For more information on using iCAP B.07.00 with vPars A.04, “CPU: Using iCAP (Instant Capacity on Demand) with vPars (vPars A.04 and iCAP B.07)” on page 188.

  — If you are using WLM, iCOD, and vPars A.03.02 or earlier together, please see the *HP-UX Virtual Partitions Ordering and Configuration Guide* for information on software version requirements.

  — Beginning with vPars A.02.01 and iCOD B.05.00, iCOD is now supported in a vPars environment.

  — Beginning with vPars A.02.01, PPU Percent Utilization is now supported in a vPars environment.

  — For iCOD/PPU versions prior to version B.05.00, vPars can be installed successfully only if *all* CPUs in the server are activated (in other words, purchased). Likewise, on a vPars server, iCOD/PPU will not load or run with versions prior to B.05.00.

  — To determine your current version of iCAP, iCOD, or vPars, use the command `swlist`.

  For information on iCOD and `LPMC_Monitor`, please see the note “Support Tools” on page 23.

- **Workload Manager (WLM)**

  If you are using WLM, iCOD, and vPars together, please see the *HP-UX Virtual Partitions Ordering and Configuration Guide* at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions for information on software version requirements.

  A virtual partition can be resized automatically based on application performance using WLM. The *Sizing vPars Automatically with HP-UX Workload Manager* white paper provides details on how to use WLM to accomplish this. The white paper can be found at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions as well as in the information library section of the WLM web page at http://www.hp.com/go/wlm.

- **Support Tools**

  For information on the required version of the Support Tools package that can run on your vPars server, see the section on Online Diagnostics in the *HP-UX Virtual Partitions Ordering and Configuration Guide*.

  Prior to STM version `A.43.00` (December 2003), in a vPars environment if the LPMC (Low Priority Machine Check) Monitor (now known as CPU Monitor) of the Support Tools bundle deactivates a processor, it does not automatically replace the failing processor with an iCAP (formerly known as iCOD) processor. The processor replacement must be performed manually using the `icod_modify` command. For more information, see the manual titled *Instant Capacity User's Guide* at http://docs.hp.com.

  Beginning with STM version A.43.00, in a vPars environment, the LPMC monitor automatically replaces a failing processor with an iCAP processor if an iCAP processor is available.

CAUTION CPU Expert Tool is supported on vPars servers. However, using this on vPars servers may cause unpredictable results. For complete information on using the Support Tools with vPars, see the Support Tools document:

http://docs.hp.com/hpux/onlinedocs/diag/stm/stm_vpar.htm

- **CSTM's CPU Info Tool**

  With STM version A.43.00, the CSTM's cpu info tool will show information about the bound processors assigned to that partition and unbound processors not assigned to any partition. For unbound procesors assigned to other partitions, it shows N/A.

  With STM version C.48.00 (HWE0505) and later, the CSTM's CPU info tool will show information about all the CPUs that are assigned to the current partition.

  When using cstm's `info` command, information is shown only for the processors that are currently owned by the virtual partition from which the `info` command is run. For the CPUs which are not assigned to the virtual partition, the information is displayed as N/A.

- **ODE Diagnostic and IO Card Utilities (Integrity Only)**

  ODE diagnostics utilities, such as CEC, CPU, or MEM, will not operate in vPars mode. Also, the IO card and diagnostic utilities, such as `FCFUPDATE` and `IODIAG.efi`, will not operate in `vPars` mode.

  You must boot the nPartition into `nPars` mode to operate these utilities.

  For more information on modes, see "Modes: Switching between nPars and vPars Modes (Integrity only)" on page 95

  Ignite-UX

  — Making Depots for Ignite-UX:

  For information on where to find a "cookbook" for setting up your Ignite-UX server for use with vPars, see "Setting Up the Ignite-UX Server" on page 64.

  — Reading the CPU counts from Ignite-UX (vPars A.03.xx and earlier):

  When Ignite-UX reports the Total Number of CPUs for a partition, it includes unassigned unbound CPUs in the count. For information on bound and unbound CPUs, see "CPU: Bound and Unbound" on page 210.

For example, if you have three virtual partitions, each with one bound CPU, and two unbound CPUs not assigned to any of the partitions, this is a total of five CPUs in the server. Ignite-UX will report three CPUs (one bound and two unbound CPUs) for each partition. However, adding up the numbers results in a total of nine CPUs for the server when there are actually only five physical CPUs.

- **(PA-RISC only) The `WINSTALL` Boot Kernel Paths with Different Versions of Ignite-UX and the `vparboot -I` command**

  The examples in this document use the Ignite-UX bootable kernel `WINSTALL` path as

  ```
  /opt/ignite/boot/WINSTALL
  ```

  This is the correct path for Ignite-UX versions B.05.xx and earlier. However, if you are using Ignite-UX version C.06.xx or later, the `WINSTALL` path has changed to

  ```
  /opt/ignite/boot/Rel_B.11.NN/?INSTALL
  ```

  where `NN` completes the HP-UX version and
  where `?` is `W` for PA-RISC systems

  For example, if vPars is using HP-UX 11i v2 (11.23) instances on a PA-RISC server with Ignite-UX version C.06.xx, the path is:

  ```
  /opt/ignite/boot/Rel_B.11.23/WINSTALL
  ```

  Thus, if you are using HP-UX 11i v2 (11.23) on a PA-RISC server and

  — Ignite-UX version B.05.xx or earlier:

  you should specify `/opt/ignite/boot/WINSTALL` on the command line (`/opt/ignite/boot` is the default); for example

  ```
  # vparboot -p <target_partition> -I <ignite_server>,/opt/ignite/boot/WINSTALL
  ```

  — Ignite-UX version C.06.xx or later:

  you must specify the absolute path `/opt/ignite/boot/Rel_B.11.23/WINSTALL` on the command line (because `/opt/ignite/boot` is the default); for example

  ```
  # vparboot -p <target_partition> -I <ignite_server>,/opt/ignite/boot/Rel_B.11.23/WINSTALL
  ```

- **Ignite-UX Recovery and Expert Recovery**

  Beginning with vPars A.02.03, the creation of `make_tape_recovery` tapes is supported on vPars-enabled servers. However, recoveries using these tapes must be done outside of the vPars environment; they cannot be used to recover a system from within a virtual partition. For example, the tape cannot be used with the `vparboot -I` command. See page 11 for a sample disaster recovery recipe that uses `make_tape_recovery`.

  Ignite-UX Recovery via `make_net_recovery` requires additional steps as noted in “Network and Tape Recovery” on page 235.

  Expert recovery works as documented in the Ignite-UX manual; however, you must account for the vPars differences described in “Expert Recovery” on page 244.

  For more information on using tape devices, see also the paper titled *Booting, Installing, Recovery, and Sharing in a vPars Environment from DVD/CDROM/TAPE/Network* available at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions.

- **Ignite-UX and other Curses Applications**

  On the virtual console, when using applications that use curses, such as the terminal versions of Ignite-UX and SAM, do *not* press `Ctrl-A` to toggle to the console display window of another virtual partition while you are still within the curses application. This is especially applicable when you are using `vparboot -I` and the Ignite-UX application to install vPars. For more information on curses, see the *curses_intro* (3X) manpage.

  As with most curses applications, if you get a garbled display, you can press `Ctrl-L` to refresh the display.

- **ServiceGuard**

  ServiceGuard is supported with vPars. However, because ServiceGuard is used to guard against hardware failures as well as software failures, its functionality will be reduced if a cluster includes multiple virtual partitions within the same server. Such configurations are not recommended. See the ServiceGuard documentation for more information on running Service Guard with vPars.

- **UPS (uninterruptible power supply) software**

  UPS hardware communicates with UPS software via the serial port. By default, a hard partition has only one serial port. For a hard partition that runs vPars, the serial port can be owned by at most one virtual partition. Therefore, on the hard partition, the UPS can communicate with only the virtual partition that owns the serial port.

  Alternately, the HP PowerTrust II-MR UPS product can be configured across virtual partitions using network connections, providing all the virtual partitions reside on the same network.

- **Processor Sets**

  vPars A.03.xx and earlier: you cannot specify a hardware path for an unbound CPU. Therefore, to avoid unintentionally removing unbound CPUs from a non-default pset, initially create the partition that will be running Processor Sets using only bound CPUs. Then, when you add or remove an unbound CPU, the unbound CPU will be added to or removed from only the default pset.

- **Glance and Openview Performance Agent (MeasureWare)**

  For correct reporting of CPU utilization, you need to run Glance and MeasureWare versions C.03.35. or higher.

- **Real-time clock (RTC)**

  Fixed in A.03.03 and later, A.04.01 and later:

  The monitor keeps track of the OS time for each virtual partition relative to the real-time clock. The OS time is the time that is changed via the `set_parms` or `date` commands.

  However, you can change the real-time clock at the `BCH` prompt or at the monitor prompt prompt (`MON>`). If you change the real-time clock, you need to run the monitor command `toddriftreset` to reset the drifts relative to the real-time clock. For information on the monitor commands, see “Using Monitor Commands” on page 118.

  Booting the machine into standalone mode from a boot disk which had its OS time ahead of the RTC will advance the RTC. If the machine is then booted into a vPars environment, the OS time of all the virtual partitions will be advanced. Administrators should ensure that the RTC is adjusted accordingly before booting the machine from standalone mode into a vPars enviroment and vice versa.

- **SCSI Initiator ID (PA-RISC only)**

  For vPars A.03.xx and earlier: the SCSI Initiator ID is the ID of the SCSI controller. Although you can display and set SCSI parameters for the SCSI controller at the BCH prompt, on a vPars server, you can also set these values from the HP-UX shell of a virtual partition using the vPars command `vparutil`. For more information, see the *vparutil* (1M) manpage.

  For vPars A.04.xx and later: please use the `mptutil` command. For information on `mptutil`, see the Ultra320 SCSI Support Guide or the support guide for your card.

- **System-wide stable storage and the `setboot` command**

  On a non-vPars server, the `setboot` command allows you to read from and write to the system-wide stable storage of non-volatile memory. However, on a vPars server, the `setboot` command does not affect the stable storage. Instead, it reads from and writes to only the partition database.

  For more information see "Boot||Shut: Setboot and System-wide Stable Storage" on page 137.

- **`mkboot` and LIF files**

  The `mkboot` command allows you to write to files in the LIF area on both Integrity and PA-RISC servers; for example, the `AUTO` file. While on a vPars server, `mkboot` can still be used to write to files in the LIF area, the LIF area is not read during the boot of the OS of a virtual partition. Instead, only the information stored in the vPars partition database is read. (Note: the files in the LIF area are still read when the system or nPartition boots).

  To simulate the effect of an AUTO file for a virtual partition, use the vPars commands so that the information is saved in the vPars partition database. For more information, see "The AUTO File on a Virtual Partition" on page 143.

- **`/stand` filesystem size**

  Due to the vPars files that will exist in `/stand`, you should increase by 100 MB the size of the `/stand` file system that you normally create.

- **`shutdown` and `reboot` commands**

  In a virtual partition, the `shutdown` and `reboot` commands shutdown and reboot a virtual partition and *not* the entire nPartition.

  Also, if a virtual partition is not set for autoboot using the autoboot attribute (see the *vparmodify* (1M) manpage), the `-r` and `-R` options of the `shutdown` or `reboot` commands will only shut down the virtual partition; the virtual partition will *not reboot*. In other words, the virtual partition will halt when the autoboot attribute is not set. For more information, see the *vparmodify* (1M) manpage.

  For the `-R` and `-r` options of the `shutdown` and `reboot` commands, the virtual partition will not reboot when there is a pending reboot for reconfiguration (RFR) until all the virtual partitions within the nPartition have been shutdown and the vPars Monitor has been rebooted. Also, the requested reconfiguration will not take place until all the virtual partitions within the involved nPartition have been shutdown and the vPars Monitor has been rebooted.

  For more information, see "Boot||Shut: Shutting Down or Rebooting a Virtual Partition" on page 133 and "Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)" on page 135.

- **`ioscan` output**

  On a PA-RISC system, the `ioscan` output for `vcn` and `vcs` drivers show a value of `NO_HW` in the `S/W State` column. This is normal.

- **`intctl` command**

  The `intctl` command is a HP-UX tool that allows you to manage IO interrupts among the active processors. It can be installed from the HP-UX Software Pack but should be used only by advanced administrators for performance tuning.

  If you are managing interrupts on vPars systems, please see the section “Managing IO Interrupts” on page 187.

- **kernel crash dump analyzer**

  You cannot use a kernel crash dump analyzer on Monitor dumps because vPars Monitor dumps are structured differently than kernel dumps. For more information on Monitor dumps, see “Monitor Dump Analysis Tool” on page 233.

- **top and other applications that show CPU ID**

  The CPU ID displayed by the `top` command and other applications may not be indicative of the actual the CPU index in standalone or nPars mode or of the actual hardware path. Within a virtual partition, `top` sees only the CPUs assigned to it. Possible `top` output is shown below; the CPU index is the leftmost column.

```
CPU   LOAD   USER   NICE    SYS   IDLE  BLOCK  SWAIT   INTR   SSYS
 0    0.01   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
 1    0.00   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
 2    0.01   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
 4    0.01   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
 7    0.01   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
 8    0.06   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
---   ----  -----  -----  -----  -----  -----  -----  -----  -----
avg   0.02   0.0%   0.0%   0.0% 100.0%   0.0%   0.0%   0.0%   0.0%
```

Ordering vPars

For the latest information on ordering vPars, please see the *HP-UX Virtual Partitions Release Notes*.

NOTE The free product known as VPARSBASE is obsolete and is no longer available or supported.

2 How vPars and its Components Work

This chapter covers:

- Partitioning Using vPars
- vPars Monitor and vPars Partition Database
- vPars Boot Sequence
- EFI and Integrity Notes
- Virtual Consoles and Logs
- Security

Partitioning Using vPars

To understand how vPars works, compare it to a server not using vPars. Figure 2-1 shows a 4-way HP-UX server. Without vPars, all hardware resources are dedicated to one instance of HP-UX and the applications that are running on this one instance.

Figure 2-1 Server without vPars

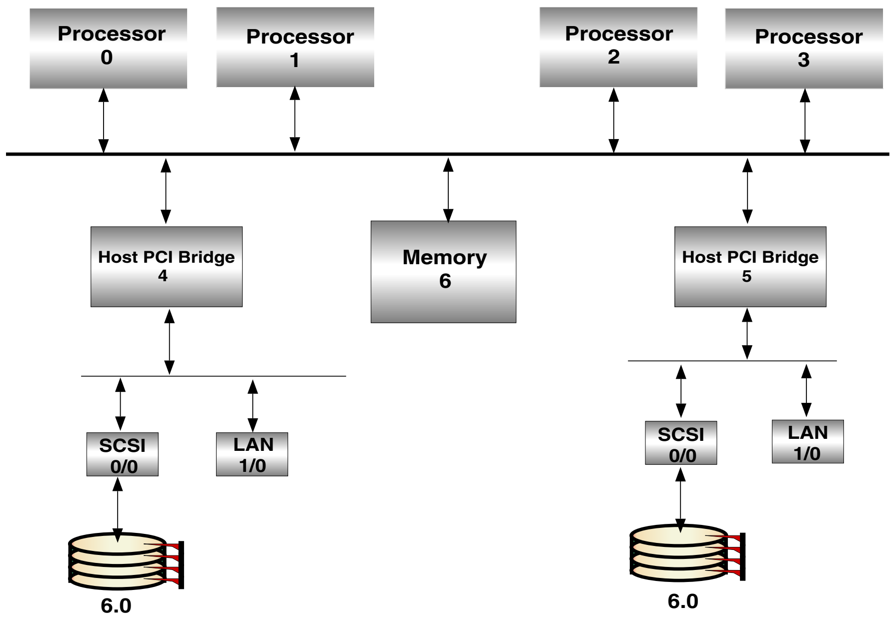
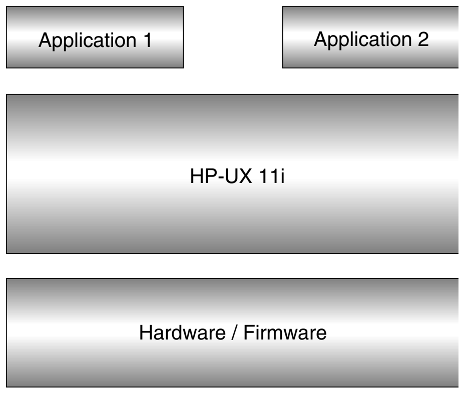


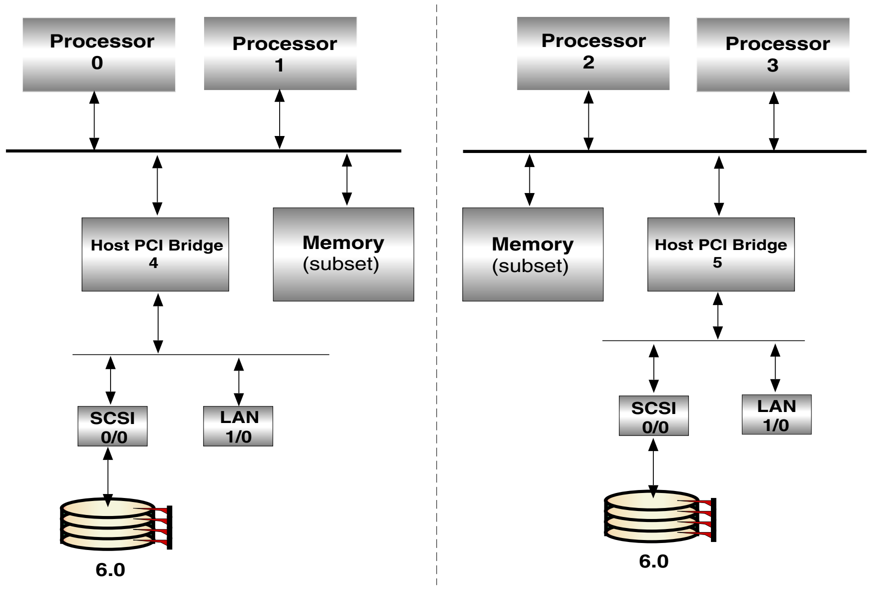
Figure 2-2 shows the software stack where all applications run on top of the single OS instance:

Figure 2-2 Software Stack of Server without vPars



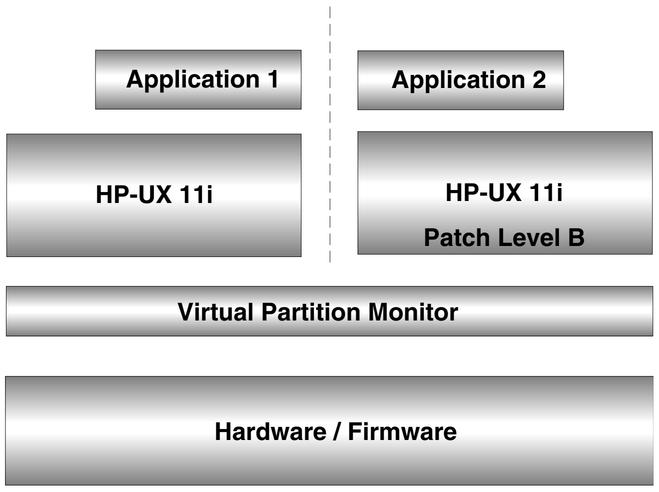
Using vPars, you can allocate a server’s resources into two or more **virtual partitions**, each with a subset of the hardware. In Figure 2-3, two virtual partitions are shown, each with its own boot disk, its own CPUs, its own LAN connection, and a sufficient subset of memory to run HP-UX and the applications intended to be hosted on that virtual partition.

Figure 2-3 Server Block Diagram with 2 Virtual Partitions



Each application can run on top of separate OS instances. Instead of a single OS instance owning all the hardware, the vPars Monitor manages the virtual partitions and their OS instances as well as the assignment of hardware resources to each virtual partition.

Figure 2-4 Software Stack for Server with 2 Virtual Partitions



vPars Monitor

For each hard partition, the vPars Monitor manages the assignment of hardware resources to virtual partitions, boots virtual partitions and their kernels, and emulates certain firmware calls. By emulating these specific calls, vPars creates the illusion to each HP-UX instance that it is running on a standalone server, consisting of the hardware that has been assigned to it.

Once a virtual partition is launched, the Monitor transfers ownership of the hardware to the virtual partition. At that point the Monitor is not involved in accessing IO hardware, physical memory, or process to processor cycles: the individual HP-UX instances have complete ownership of their respective hardware resources. This allows each partition to run at full speed.

The commands for the vPars Monitor are shown in the section “Monitor: Using Monitor Commands” on page 105; however, most of the vPars operations are performed using vPars commands at the Unix shell level. For more information on the commands, see the chapter “Monitor and Shell Commands” on page 93.

vPars Partition Database

At the heart of the vPars Monitor is the **partition database**. The partition data

base contains partition configuration information. Using the partition database, the Monitor tracks which virtual partitions exist and what hardware resources and partition attributes are associated with each partition.

When the Monitor boots (see “Boot Sequence” on page 33), it reads a copy of the partition database from a file on the same disk from which the Monitor `/stand/vpmon` is booted. The default file is `/stand/vpdb`. Then, the Monitor creates a master copy of the vPars partition database in the memory reserved by the Monitor.

The operating system of each virtual partition also keeps a local copy of the partition database in a file, by default `/stand/vpdb`, on its local boot disk.

You can create, modify, and view the database contents using vPars commands at the Unix shell level. See “Monitor and Shell Commands” on page 93. Because the format of the database is proprietary, you must use only vPars commands to create, modify, and view the database.

Whenever you execute a vPars command from the Unix shell of a partition, the change is made first to the Monitor’s master copy. Then, the operating system from which you executed the command updates its local copy from the master copy. Every five seconds, the operating system of each running partition automatically updates its local copy from the master copy. This synchronization ensures that the virtual partitions and changes to the partition database are preserved when the entire hard partition is rebooted.

NOTE The Monitor can only synchronize to the database files of running virtual partitions. If you reboot the hard partition, you should boot the Monitor from the boot disk of a virtual partition that was running during your most recent partition configuration change.

Boot Sequence

NOTE This section describes a manual boot sequence to help explain how vPars impacts the boot process, but you can continue to use an autoboot sequence to boot all partitions. See "Boot||Shut: Autoboot" on page 143.

Boot Sequence: Quick Reference

On a server without vPars, a simplified boot sequence is:

PA-RISC		**Integrity**	
1. ISL	(Initial System Loader)	1. EFI	(Extensible Firmware Interface)
2. hpux	(secondary system loader)	2. hpux.efi	
3. /stand/vmunix	(kernel)	3. /stand/vmunix	

Adding vPars adds the Monitor layer, so now `hpux` (for Integrity, `hpux.efi`) loads the Monitor. Then the Monitor boots the kernels of the virtual partitions. The boot sequence becomes

1. ISL or EFI	(firmware)
2. hpux or hpux.efi	
3. /stand/vpmon	(vPars Monitor and partition database)
4. /stand/vmunix	(kernels of the virtual partitions)

Boot Sequence: The Details

With or without vPars, the firmware loads and launches ISL or EFI.

PA-RISC	**Integrity**

`ISL>`	`Shell> fs0:`
	`fs0:\> \efi\hpux\hpux.efi`

In a server without vPars, from ISL or EFI, the loader `hpux or hpux.efi` loads the kernel `/stand/vmunix`:

PA-RISC	**Integrity**
`ISL> hpux /stand/vmunix`	`HPUX> boot vmunix`

However, in a server with vPars, from the loader `(hpux or hpux.efi)` loads the vPars Monitor `(/stand/vpmon)`:

PA-RISC	**Integrity**
`ISL> hpux /stand/vpmon`	`HPUX> boot vpmon`

The Monitor loads the partition database (the default is `/stand/vpdb`) from the same disk that `/stand/vpmon` was booted. The Monitor internally creates (but does not boot) each virtual partition according to the resource assignments in the partition database.

Next, the vPars Monitor runs in interactive mode (when no options to `/stand/vpmon` are given) with a command line interface.

```
MON>
```

To boot a kernel in a virtual partition (that is, to launch a virtual partition), use the Monitor command `vparload`. For example, to launch the virtual partition named `uma1`:

```
MON> vparload -p uma1
```

In this example, the vPars Monitor would load the virtual partition uma1 and launch the kernel from the boot device specified for uma1. (The boot device is assigned when the virtual partition is created and is recorded in the Monitor database.)

HP-UX is now booted on the virtual partition `uma1`.

Once a virtual partition is running, you will be at the virtual console of a virtual partition. Subsequent virtual partitions can be booted using the vPars command `vparboot` at the HP-UX shell prompt of `uma1`.

For more information on the HP-UX boot process, see the following manpages:

boot (1M)
efi (4)
hpux (1M)
hpux.efi (1M)
isl (1M)
pdc (1M)
setboot (1M)

For more information on how to boot a virtual partition, see “Boot||Shut: Booting a Virtual Partition” on page 132.

EFI and Integrity Notes

- **EFI Shell Accessibility**

  After the vPars Monitor (`/stand/vpmon`) is booted, the EFI shell will not be accessible. This includes using `hpux.efi` and other EFI commands.

  If you need to perform any EFI functions, you will need to shut down all the virtual partitions and reboot the nPartition to access the EFI shell.

- **New vPars Commands**

  The vPars commands introduced in vPars A.04.01 for use on **only Integrity systems** are `vparenv`, `vparconfig`, and `vparefiutil`:

`vparenv`	**Unix shell** command that allows you to **set the mode** (vPars or nPars) for the next reboot of the nPartition or to set the memory granularity unit size in firmware
`vparconfig`	**EFI command** that allows you to **set the mode** (`vPars` or `nPars`) and forces a reboot of the nPartition. Note that `vparconfig` is **not a built-in EFI command**; you will need to go to the `fsN:\>` disk prompt to execute this command. `vparconfig` is installed in the EFI partition of the root disk when vPars is installed. Specifically, the file is `vparconfig.efi` and is installed in `\efi\hpux`.
`vparefiutil`	**Unix shell command** to display or manage the **HP-UX hardware path to EFI path mappings** of bootable disks within the vPars database.

  When booting the Monitor from EFI (boot /stand/vpmon), the backspace key sometimes is not parsed correctly; if the command fails, try again without backspacing.

  For more information on:

  — using `vparenv` or `vparconfig` to switch modes, see “Modes: Switching between nPars and vPars Modes (Integrity only)” on page 95.

  — using `vparenv` and granularity, see “Memory: Setting Granularity Values” on page 172.

  — using `vparefiutil`, see “EFI Boot Disk Paths, including Disk Mirrors, and vparefiutil (Integrity only)” on page 100.

- **CPUs and Deconfiguration**

  If a CPU is **marked for deconfiguration** using an EFI command and the nPartition is not rebooted (for example, the vPars Monitor is immediately booted), the vPars Monitor will not know or indicate (including with vparstatus) that the CPU has been marked for deconfiguration and will use the CPU like any other working CPU.

- **EFI Variables and Switching Modes**

  The default EFI settings in `nPars` mode will be inherited when switched to `vPars` mode. However, when switching back to `nPars` mode, any **EFI settings will be reset to the nPartition defaults**., unless otherwise noted (for example, memory granularity). This **incudes the primary and alternate paths** (HAA (High-Availability Alternate) is not supported). Even if you use `parmodify` to change the paths. `parstatus` will show them as set; however, once the system is booted into nPars mode, those changes by `parmodify` are not retained. For more information on switching modes, see the manpage *vparenv* (1M).

Also, while running in vPars mode, the EFI device path of a boot device, specifically the Monitor boot device, can be changed when the boot device is reformatted due to an installation (either cold or Ignite-UX). The associated EFI boot path is updated to use the new EFI device path. However, firmware-saved EFI boot path options are not updated. If the nPartition or Monitor is rebooted, the new EFI boot path options are discarded and replaced with the previously saved EFI boot path options, which contain now stale EFI device paths.

The EFI boot options can be updated manually by performing the following:

1. Switch to `nPars` mode via the `vparenv` or `vparconfig` command.
2. Reboot the nPartition to the EFI Boot Manager.
3. Select `Boot Option Maintenance Menu`.
4. Select `Delete Boot Option(s)`.
5. Select `HP-UX Primary Boot` and then exit.
6. Select `Exit` to return to the EFI Boot Manager.
7. Select `EFI Shell [Built-in]`.
8. Launch HP-UX from the EFI shell prompt:

   ```
   Shell> fsN:
   ```

   ```
   fsN:\> efi\hpux\hpux boot vmunix
   ```

9. Use the `setboot` command to set up the primary boot path with the desired boot device. This also sets the `HP-UX Primary Boot` boot option with the latest EFI device path.
10. Use `the vparenv` command to switch to `vPars` mode
11. Reboot the nPartition.

- **EFI command `default clear`**

  Whenever you use the `default clear` command at the EFI shell, this erases vPars information that is stored in NVRAM and the vPars monitor may not boot. To boot the monitor, you should perform the following:

  1. Change the mode to `nPars` and allow the system to reboot to the EFI shell:

     ```
     Shell> fsN:
     fsN:\> efi\hpux\vparconfig reboot nPars
     ```

  2. At the EFI prompt, boot the HP-UX kernel in standalone mode:

     ```
     Shell> fsN:
     fsN:\> efi\hpux\hpux /stand/vmunix
     ```

     The booting of the kernel will restore the vPars information in NVRAM. Now you can return to vPars mode and reboot the Monitor.

  3. To change the mode to `vPars` and reboot the Monitor:

     ```
     # vparenv -m vPars
     # shutdown -r
     ...

     Shell> fs0:
     fs0:\> \efi\hpux\hpux vpmon
     ```

Integrity Differences Relative to PA-RISC

Beginning with vPars A.04.01, vPars is supported on both Integrity and PA-RISC platforms. This section describes the major conceptual differences for booting and running vPars on Integrity relative to PA-RISC.

Booting

- **Modes**

  On Integrity platforms, you have to set the mode (vPars or nPars) to be able to boot the nPartition into standalone (nPars) or the vPars environment (vPars).

  See “Modes: Switching between nPars and vPars Modes (Integrity only)” on page 95.

  On PA-RISC, you do not need to set modes.

- **vparboot -I and the LAN card**

  On Integrity platforms, performing a vparboot -I uses the lan card of the target partition to obtain the bootable kernel.

  See “Ignite-UX, the LAN, the LAN card, and vparboot -I” on page 65.

  On PA-RISC, the lan card of the source partition is used.

- **Boot string**

  On Integrity platforms, the boot string used at the hpux.efi prompt (hpux>) is “boot vpmon”.

  See “Boot Sequence” on page 33.

  See “Monitor: Booting the vPars Monitor” on page 103.

  See “Boot||Shut: Autoboot” on page 143.

  On PA-RISC, the boot string at the hpux prompt (HPUX>) is “hpux /stand/vpmon”.

Commands

- **vPars commands**

  These commands are effective only on Integrity:

  — `vparefiutil`
  — `vparenv`
  — `vparconfig`

  See “EFI and Integrity Notes” on page 35.

  These commands are effective only on PA-RISC:

  — `vparreloc`
  — `vparutil`

See also:

- “EFI and Integrity Notes” on page 35
- “Feature Differences” on page 224

Virtual Consoles

HP-UX servers have a special terminal or window called a console that allows special control and displays system error messages.

With vPars, each virtual partition has its own **virtual console**. On Integrity, the console is virtualized by firmware (and therefore, there is no vcs driver). On PA-RISC, for each partition, its console IO is sent to its vcn (Virtual CoNsole) driver. From the vcn driver, the console IO is sent to the Monitor. From the Monitor, the console IO is sent to the vcs (virtual console slave) driver of the partition that owns the hardware console port. Finally, the vcs driver sends the console IO to the physical hardware console. It is this vcs driver that manages the console IO to the actual hardware console port.

When the partition that owns the hardware console port is not running, the vPars Monitor takes over the management of the IO to the hardware console port, so you will still have access to the virtual console displays.

You can access the console port as you would on any non-vPars server, for example, through a dumb terminal or lan console. Then, to cycle between the virtual console displays of the various partitions, press `Ctrl-A`.

Each virtual partition has an 8K circular buffer for console output. If not already displayed, the Monitor copies this 8K buffer to the console when you press `Ctrl-A`.

CAUTION (PA-RISC only) The first virtual partition that you create must own the LBA (local bus adapter) that contains the physical hardware console port. For an example, see “Ensuring the Hardware Console Port Is Owned by the First Virtual Partition (PA-RISC)” on page 55.

NOTE Note the following when using virtual consoles:

- **`ioscan` output**

  On a PA-RISC system, the ioscan output for `vcn` and `vcs` drivers show a value of `NO_HW` in the `S/W State` column. This is normal.

  On an Integrity system, the vPars virtual console is truly virtual and will not show up in an `ioscan`. You can see this with the `vparstatus -m` command:

```
# vparstatus -m
Console path:  No path as console is virtual
Monitor boot disk path:  13.0.11.1.0.8.0
Monitor boot filename:  /stand/vpmon
Database filename:  /stand/vpdb
Memory ranges used:  0x0/232611840 Monitor
                     0xddd6000/688128 firmware
                     0xde7e000/1384448 Monitor
                     0xdfd0000/33751040 firmware
                     0x10000000/134213632 Monitor
                     0x7fffe000/8192 firmware
                     0x8a0ff000000/16777216 firmware
```

- Potential for Lost Output

  Because the console output is a circular buffer, output beyond the 8K is overwritten and lost.

- Active Console IO when Multiple Virtual Partitions are Booted

It is not deterministic which virtual partition will be active with the physical console when multiple virtual partitions are booted.

- Switchover Pause with Shutting Down

  When the virtual partition that owns the hardware console port is shut down, there will be a pause of console output (the system is *not* hung) as console IO management switches over from the virtual partition to the vPars Monitor. Console output resumes automatically after the pause. You will not lose any console output. During the switchover period, no console input is accepted.

  For rp7400/N4000 and rp5470/L3000 servers, the pause can be from ten to twenty seconds. For Superdome and other nPartitionable servers, the switchover pause can be minutes, depending on the amount of memory owned by the virtual partition that owns the hardware console port.

- Pause when Booting from Tape

  The system may appear but is actually not hung when booting from tape due to the increased time it takes to load a kernel from tape instead of from disk.

- Switchover Pause during the Crash State

  Whenever the virtual partition that owns the hardware console port is in the `crash` state, the switchover pause will occur and remain as long as the virtual partition is in this `crash` state. For more information on the `crash` state, see the *vparstatus* (1M) manpage and “Commands: Displaying Monitor and Resource Information (vparstatus)” on page 114.

- GSPdiag1 device file

  The GSPdiag1 device file (`/dev/GSPdiag1`) can only be accessed from the virtual partition that contains the console hardware port.

- Terminal Emulation

  To avoid display problems, be sure that the terminal setting of the GSP on the vPars server matches the terminal or terminal emulator that you are using to access it. For details on how to do this, see “Setting the GSP Terminal Type” on page 69.

- Ignored Keyboard Input (A.03.xx only)

  There is one known case where the virtual console will ignore keyboard input (data sent to the console continues to be displayed; only keyboard input is ignored). This occurs when the virtual partition that owns the hardware console port is down and the CPU with the lowest hardware path is not assigned to any virtual partition. When this CPU is migrated to a running virtual partition, the console will not accept any keyboard input.

  You can do either of the following to resolve the problem:

  — From a running partition, reset the partition that owns the hardware console port by executing `vparreset -p target_partition -h`, where `target_partition` is the partition that owns the hardware console port.

  — From a running partition, boot the partition that owns the hardware console port by executing `vparboot -p target_partition`, where `target_partition` is the partition that owns the hardware console port

  If no other virtual partitions are accessible, you must reboot the server or nPartition in order to regain console input.

- Toggling Past the Monitor Prompt (A.03.xx only)

When the monarch CPU of the server is not assigned to any partition, you will see the Monitor prompt. Press `Ctrl-A` to cycle to the console window of the next partition.

nPartition Logs

On an nPartition server running vPars, all virtual partitions within an nPartition share the same console device: the **nPartition's console**. Thus, an nPartition's console log contains console IO for multiple virtual partitions. Further, since the vPars Monitor interface is displayed and accessed through the nPartition's console, vPars Monitor output is also recorded in the nPartition's console log. There is only one Monitor per nPartition.

The **server chassis logs** record nPartition and server complex hardware events. The chassis logs do not record vPars-related configuration or vPars boot events (PA-RISC only); however, the chassis logs do record HP-UX "heartbeat" events. The server chassis logs are viewable from the GSPs Show Chassis Log menu. For more information, see the Help within the GSPs online help.

The **vPars Monitor event logs** record only vPars events; it does not contain any nPartition chassis events. For more information, see *vparstatus* (1M).

Also, for a given nPartition, the Virtual Front Panel (VFP) of the nPartition's console displays an OS heartbeat whenever at least one virtual partition within the nPartition is up.

MCA (Machine Check Abort) Logs on Integrity Systems

Description

An **MCA** is a processor interrupt that occurs when the processor discovers that it can not continue reliable operation. An MCA can result from either a hardware problem (such as an uncorrectable data error in memory or on a system bus) or from a software error (typically, in a driver). In most cases when an MCA occurs, the system stops normal processing and takes an OS memory dump if possible. The firmware also automatically logs data that can be used by HP tools to analyze the cause of the MCA. On reboot, this data is read from firmware and saved in “MCA logs”.

Two different types of MCAs can occur. On an Integrity nPartition running vPars, the first type will only affect one virtual partition and is called a “local MCA”. The second type will affect all the virtual partitions in an nPartition and is called a “Global MCA”.

Location of Log Files

On an nPartition not running vPars, the MCA logs are gathered from the firmware during OS reboot and saved in the /var/tombstones directory. Typically, multiple files are created of the form mca*.

When running vPars, logs from a local MCA are saved in the virtual partition that experienced the MCA. Similar to the non-vPars configuration, these files are in the /var/tombstones directory of the virtual partition. Logs from a global MCA are saved in the /var/tombstones directory of only one particular virtual partition. The virtual partition that is used is the virtual partition that was booted from the *same disk that was used to boot the vPars Monitor*; this disk must be the primary boot disk specified in the EFI Boot Manager after the system reboots in vPars mode following an MCA.

Security

You should be aware of the following security concerns:

- The vPars commands (as described in “Monitor and Shell Commands” on page 93) are restricted to root access, but the commands work on any of the virtual partitions, regardless of which partition the commands are executed from. Therefore, a user on one partition can affect another virtual partition by targeting the virtual partition in a vPars command. For example, a root user running on the partition `vpar2` can reset the partition `vpar3` using the `vparreset` command.
- A user with console access can gain access to the file systems on any of the virtual partitions in the hard partition.

NOTE Additional Security Functionality

- A.04.xx:

  A white paper on using RBAC (Role-based Access Control) with vPars A.04.xx is available on docs.hp.com.
- A.03.03:

  A new feature called Primary-Admin Virtual Partitions is available. For more information, see the Chapter 10, “vPars Flexible Administrative Capability (vPars A.03.03 and vPars A.04.02),” on page 245.

3 Planning Your System for Virtual Partitions

This chapter covers

- Example System
- Planning Your Virtual Partitions
- Hardware Path Formats

full ioscan output of non-cellular system named winona

```
winona# ioscan

H/W Path      Class                     Description
===================================================
              root
0             ioa                   System Bus Adapter (803)
0/0               ba                    Local PCI Bus Adapter (782)
0/0/0/0                 lan             HP PCI 10/100Base-TX Core
0/0/1/0                 ext_bus         SCSI C895 Fast Wide LVD
0/0/1/0.7                  target
0/0/1/0.7.0                   ctl       Initiator
0/0/2/0                 ext_bus         SCSI C875 Ultra Wide Single-Ended
0/0/2/0.6                  target
0/0/2/0.6.0                   disk      SEAGATE ST39102LC
0/0/2/0.7                  target
0/0/2/0.7.0                   ctl       Initiator
0/0/2/1                 ext_bus         SCSI C875 Ultra Wide Single-Ended
0/0/2/1.7                  target
0/0/2/1.7.0                   ctl       Initiator
0/0/4/0                 tty             PCI Serial (103c1048)
0/0/5/0                 tty             PCI Serial (103c1048)
0/1               ba                    Local PCI Bus Adapter (782)
0/2               ba                    Local PCI Bus Adapter (782)
0/4               ba                    Local PCI Bus Adapter (782)
0/4/0/0                 ba              PCItoPCI Bridge
0/4/0/0/4/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/4/0/0/5/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/4/0/0/6/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/4/0/0/7/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/5               ba                    Local PCI Bus Adapter (782)
0/5/0/0                 ba              PCItoPCI Bridge
0/5/0/0/4/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/5/0/0/5/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/5/0/0/6/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/5/0/0/7/0                   lan       HP A5506A PCI 10/100Base-TX 4 Port
0/8               ba                    Local PCI Bus Adapter (782)
0/8/0/0                 ext_bus         SCSI C875 Fast Wide Differential
0/8/0/0.5                  target
0/8/0/0.5.0                   disk      SEAGATE ST39175LC
0/8/0/0.7                  target
0/8/0/0.7.0                   ctl       Initiator
0/8/0/1                 ext_bus         SCSI C875 Fast Wide Differential
0/8/0/1.7                  target
0/8/0/1.7.0                   ctl       Initiator
0/10              ba                    Local PCI Bus Adapter (782)
0/12              ba                    Local PCI Bus Adapter (782)
1             ioa                       System Bus Adapter (803)
1/0               ba                    Local PCI Bus Adapter (782)
1/2               ba                    Local PCI Bus Adapter (782)
1/2/0/0                 ext_bus         SCSI C875 Fast Wide Differential
1/2/0/0.0                  target
1/2/0/0.0.0                   disk      SEAGATE ST39102LC
1/2/0/0.7                  target
1/2/0/0.7.0                   ctl       Initiator
1/2/0/1                 ext_bus         SCSI C875 Fast Wide Differential
1/2/0/1.7                  target
1/2/0/1.7.0                   ctl       Initiator
1/4               ba                    Local PCI Bus Adapter (782)
1/4/0/0                 ext_bus         SCSI C875 Fast Wide Differential
```

```
1/4/0/0.5                  target
1/4/0/0.5.0                   disk      SEAGATE ST39175LC
1/4/0/0.7                  target
1/4/0/0.7.0                   ctl       Initiator
1/4/0/1                 ext_bus         SCSI C875 Fast Wide Differential
1/4/0/1.7                  target
1/4/0/1.7.0                   ctl       Initiator
1/8               ba                    Local PCI Bus Adapter (782)
1/10              ba                    Local PCI Bus Adapter (782)
1/10/0/0                ba              PCItoPCI Bridge
1/10/0/0/4/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/10/0/0/5/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/10/0/0/6/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/10/0/0/7/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/12              ba                    Local PCI Bus Adapter (782)
1/12/0/0                ba              PCItoPCI Bridge
1/12/0/0/4/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/12/0/0/5/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/12/0/0/6/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
1/12/0/0/7/0                  lan       HP A5506A PCI 10/100Base-TX 4 Port
32            pbc                       Bus Converter
33            processor                 Processor
36            pbc                       Bus Converter
37            processor                 Processor
40            pbc                       Bus Converter
41            processor                 Processor
44            pbc                       Bus Converter
45            processor                 Processor
96            pbc                       Bus Converter
97            processor                 Processor
100           pbc                       Bus Converter
101           processor                 Processor
104           pbc                       Bus Converter
105           processor                 Processor
108           pbc                       Bus Converter
109           processor                 Processor
192           memory                    Memory
```

full ioscan output of cellular (nPartitionable) system named keira

```
keira# ioscan
H/W Path        Class                                          Description
===========================================================================
                root
0               cell
0/0                 ioa                                        System Bus Adapter (805)
0/0/0                  ba                                      Local PCI Bus Adapter (782)
0/0/0/0/0                    tty                               PCI SimpleComm (103c1290)
0/0/0/0/1                    tty                               PCI Serial (103c1048)
0/0/0/3/0                    ext_bus                           SCSI C1010 Ultra160 Wide LVD A6793-60001
0/0/0/3/0.6                     target
0/0/0/3/0.6.0                      disk                        HP 36.4GST336753LC
0/0/0/3/0.7                     target
0/0/0/3/0.7.0                      ctl                         Initiator
0/0/0/3/1                    ext_bus                           SCSI C1010 Ultra Wide  A6793-60001
0/0/0/3/1.6                     target
0/0/0/3/1.6.0                      ctl                         Initiator
0/0/1                  ba                                      Local PCI-X Bus Adapter (783)
0/0/1/1/0                    ba                                PCItoPCI Bridge
0/0/1/1/0/4/0                      lan                         HP A5506B PCI 10/100Base-TX 4 Port
0/0/1/1/0/5/0                      lan                         HP A5506B PCI 10/100Base-TX 4 Port
0/0/1/1/0/6/0                      lan                         HP A5506B PCI 10/100Base-TX 4 Port
0/0/1/1/0/7/0                      lan                         HP A5506B PCI 10/100Base-TX 4 Port
0/0/2                  ba                                      Local PCI-X Bus Adapter (783)
0/0/2/1/0                    lan                               HP A6825-60101 PCI 1000Base-T Adapter
0/0/4                  ba                                      Local PCI-X Bus Adapter (783)
0/0/6                  ba                                      Local PCI-X Bus Adapter (783)
0/0/6/1/0                    fc                                HP Tachyon XL2 Fibre Channel Mass Storage
Adapter
0/0/8                  ba                                      Local PCI-X Bus Adapter (783)
0/0/8/1/0                    ba                                PCItoPCI Bridge
0/0/8/1/0/1/0                      ext_bus                     SCSI C1010 Ultra160 Wide LVD
0/0/8/1/0/1/0.7                                    target
0/0/8/1/0/1/0.7.0                                     ctl                   Initiator
0/0/8/1/0/1/1                      ext_bus                     SCSI C1010 Ultra160 Wide LVD
0/0/8/1/0/1/1.7                                    target
0/0/8/1/0/1/1.7.0                                     ctl                   Initiator
0/0/8/1/0/4/0                      lan                         HP A6794-60001 PCI 1000Base-T
0/0/10                 ba                                      Local PCI-X Bus Adapter (783)
0/0/12                 ba                                      Local PCI-X Bus Adapter (783)
0/0/14                 ba                                      Local PCI-X Bus Adapter (783)
0/5                 memory                                     Memory
0/10                processor                                  Processor
0/11                processor                                  Processor
0/12                processor                                  Processor
0/13                processor                                  Processor
0/14                processor                                  Processor
0/15                processor                                  Processor
1               cell
1/0                 ioa                                        System Bus Adapter (805)
1/0/0                  ba                                      Local PCI Bus Adapter (782)
1/0/0/0/0                    tty                               PCI SimpleComm (103c1290)
1/0/0/0/1                    tty                               PCI Serial (103c1048)
1/0/0/3/0                    ext_bus                           SCSI C1010 Ultra160 Wide LVD A6793-60001
1/0/0/3/0.6                     target
1/0/0/3/0.6.0                      disk                        HP 36.4GST336753LC
1/0/0/3/0.7                     target
1/0/0/3/0.7.0                      ctl                         Initiator
1/0/0/3/1                   ext_bus                         SCSI C1010 Ultra Wide Single-Ended A6793-60001
```

```
1/0/0/3/1.7                     target
1/0/0/3/1.7.0                      ctl                        Initiator
1/0/1                  ba                                     Local PCI-X Bus Adapter (783)
1/0/1/1/0                    ba                               PCItoPCI Bridge
1/0/1/1/0/1/0                      ext_bus                    SCSI C1010 Ultra160 Wide LVD
1/0/1/1/0/1/0.7                                   target
1/0/1/1/0/1/0.7.0                                    ctl                   Initiator
1/0/1/1/0/1/1                      ext_bus                    SCSI C1010 Ultra160 Wide LVD
1/0/1/1/0/1/1.7                                   target
1/0/1/1/0/1/1.7.0                                    ctl                   Initiator
1/0/1/1/0/4/0                      lan                        HP A6794-60001 PCI 1000Base-T
1/0/2                  ba                                     Local PCI-X Bus Adapter (783)
1/0/4                  ba                                     Local PCI-X Bus Adapter (783)
1/0/4/1/0                    ba                               PCItoPCI Bridge
1/0/4/1/0/4/0                     fc                       HP 2 GB PCI/PCI-X Fibre Channel FC/GigE Dual
Port Combo Adapter
1/0/4/1/0/4/0.1                                   fcp                      FCP Domain
1/0/4/1/0/4/0.1.0.0.0                                      ext_bus         FCP Array Interface
1/0/4/1/0/4/0.1.0.0.0.0                                       target
1/0/4/1/0/4/0.1.0.0.0.0.1                                        disk      COMPAQ  MSA1000 VOLUME
1/0/4/1/0/4/0.1.0.255.0                                    ext_bus         FCP Device Interface
1/0/4/1/0/4/0.1.0.255.0.0                                     target
1/0/4/1/0/4/0.1.0.255.0.0.0                                      ctl       COMPAQ  MSA1000
1/0/4/1/0/6/0                     lan                       HP A9784-60001 PCI/PCI-X 1000Base-T FC/GigE
Combo Adapter
1/0/6                  ba                                     Local PCI-X Bus Adapter (783)
1/0/8                  ba                                     Local PCI-X Bus Adapter (783)
1/0/10                 ba                                     Local PCI-X Bus Adapter (783)
1/0/12                 ba                                     Local PCI-X Bus Adapter (783)
1/0/12/1/0                   ext_bus                          SCSI C1010 Ultra160 Wide LVD A6829-60101
1/0/12/1/0.7                    target
1/0/12/1/0.7.0                     ctl                        Initiator
1/0/12/1/0.8                    target
1/0/12/1/0.8.0                     disk                       HP 36.4GST336753LC
1/0/12/1/1                   ext_bus                          SCSI C1010 Ultra160 Wide LVD A6829-60101
1/0/12/1/1.7                    target
1/0/12/1/1.7.0                     ctl                        Initiator
1/0/14                 ba                                     Local PCI-X Bus Adapter (783)
1/5                 memory                                    Memory
1/6                 ipmi                                      IPMI Controller
1/10                processor                                 Processor
1/11                processor                                 Processor
1/12                processor                                 Processor
1/13                processor                                 Processor
1/14                processor                                 Processor
1/15                processor                                 Processor
```

Planning, Installing, and Using vPars with an nPartitionable Server

When using vPars, the major difference between non-nPartitionable and nPartitionable systems is the hardware path.

IO Hardware Paths

For non-nPartitionable systems, the beginning portions of the IO hardware paths are in the format:

```
sba/lba
```

But for nPartitionable systems, the beginning portions of the IO hardware paths include the cell and are in the format:

```
cell/sba/lba
```

Impact on vPars Commands: Specifying IO

On a non-nPartitionable system, a `vparcreate` command might look like:

```
# vparcreate -p winona1 -a cpu::2 -a cpu:::2 -a mem::1024 -a io:0.0 -a io:0.4 -a
io:0.0.2.0.6.0:BOOT
```

where `-a io:0.0` represents the `sba/lba` format.

But on an nPartitionable system, the equivalent `vparcreate` command would look like:

```
# vparcreate -p vpar1 -a cpu::2 -a cpu:::2 -a mem::1024 -a io:0.0.0 -a io:0.0.4 -a
io:0.0.0.2.0.6.0:BOOT
```

where the `-a io:0.0.0` represents the `cell/sba/lba` format. If only `-a io:0.0` were used on an nPartitionable system, this would be specifying only the `cell/sba`.

CAUTION When using vPars A.03.01 or earlier, IO is assigned only at or below the LBA level. For correct IO allocation, you must *include the LBA*. *Specifying only the SBA is not supported*. On nPartitionable systems, if you specify only the `cell/sba` format for IO allocation, the vPars commands will not assume that all LBAs under the SBA are to be included in the allocation; the system may panic.

NOTE When specifying the boot disk or alternate boot disk hardware paths, the *full hardware path* must always be specified.

CPU Hardware Paths

The same is true for CPU hardware paths. In the non-nPartitionable systems, the CPU path is

```
cpu
```

But for nPartitionable systems, the CPU path includes the cell, so the CPU path is

```
cell/cpu
```

Impact on vPars Commands: Specifying CPU

Since the nPartitionable systems include the cell in the hardware path, when specifying a CPU hardware path, you must include the cell number to specify the entire CPU hardware path.

On a non-nPartitionable system, if the `ioscan` output shows

```
41 processor Processor
45 processor Processor
```

where `41` and `45` are the hardware paths of two CPUs. Thus, the `vparcreate` command might look like:

```
# vparcreate -p winona2 -a cpu::2 -a cpu:::2 -a cpu:41 -a cpu:45 ...
```

But for an nPartitionable system, if the `ioscan` output shows

```
0/12 processor Processor
0/13 processor Processor
```

where `0/12` and `0/13` are the cell/processor hardware paths, then the `vparcreate` command would look like:

```
# vparcreate -p vpar2 -a cpu::2 -a cpu:::2 -a cpu:0/12 -a cpu:0/13 ...
```

Planning Your Virtual Partitions

Virtual Partitions Layout Plan

Before you install vPars, you should have a plan of how you want to configure the virtual partitions within your server.

Example of a virtual partition plan for vPars A.04.xx based on the example cellular server:

Partition Name	keira1	keira2	keira3

Assigned CPUs (A.04.xx)	`num = 2`	`num = 1 and`
		`1 from cell 1`
			`num = 1`

Unassigned CPUs (A.04.xx)	`three CPUs are available`		
Memory	`1024 MB`	`1024 MB`	`1024 MB`

IO LBAs	`1.0.0`
	`0.0.1`
		`1.0.4`
		`1.0.1`
			`0.0.0`
			`0.0.2`

Boot Path	`1/0/0/3/0.6.0.0.6.0`	`1/0/4/1/0/4/0.1.0.0.0.0.1`	`0/0/0/3/0.6.0`
LAN	`0/0/1/1/0/4/0`	`1/0/1/1/0/4/0`	`0/0/2/1/0`
console port (PA-RISC Only)	`owned by keira1`		
Autoboot	`AUTO`	`MANUAL`	`AUTO`

Example of a virtual partition plan for vPars A.03.xx based on the example non-cellular server:

Partition Name	winona1	winona2	winona3

Bound CPUs (A.03.xx)	`total = 2`
	`min   = 2`
		`total = 2`
		`min   = 2`
		`paths = 41,45`
			`total = 1`
			`min   = 1`

Unbound CPUs (A.03.xx)	`three CPUs are available`		
Memory	`1024 MB`	`1280 MB`	`1280 MB`

IO LBAs	`0.0`
	`0.4`
		`0.8`
		`1.10`
			`0.5`
			`1.4`

Boot Path	`0.0.2.0.6.0`	`0.8.0.0.5.0`	`1.4.0.0.5.0`
LAN	`0.0.0.0`	`1.10.0.0.4.0`	`0.5.0.0.4.0`

console port (PA-RISC Only)	owned by winona1		
Autoboot	AUTO	AUTO	AUTO

NOTE When you create a partition, the vPars Monitor assumes you will boot and use the partition. Therefore, even if a partition is down, the resources assigned to the partition cannot be used by any other partition.

The next few sections will describe how we arrived at each portion of the partition plan.

Number of Virtual Partitions

For the latest information on the recommended and maximum number of virtual partitions per system or nPartition, please see the document *HP-UX Virtual Partitions Ordering and Configuration Guide* available at:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

Virtual Partition Names

All virtual partitions must be given text names that are used by the vPars commands. The names can consists of only alphanumeric characters and periods ('.'). The maximum length of a name is 239 characters.

HP recommends using the corresponding hostnames for virtual partition names, but they are not internally related.

For our cellular server, we have chosen the names of our virtual partitions to be `keira1`, `keira2`, and `keira3`:

Partition Name	keira1	keira2	keira3

For our non-cellular server, we have chosen the names of our virtual partitions to be `winona1`, `winona2`, and `winona3`:

Partition Name	winona1	winona2	winona3

Although the underscore (_) is a legal character within the name of a virtual partition, it is not a legal character within the Domain Name System (DNS).

TIP Virtual Partitions on nPartitions

If you are using vPars on a complex, you may want to distinguish the names of your virtual partitions from the names of your nPartitions to avoid confusion.

Minimal Hardware Configuration

Every bootable virtual partition must have at least:

- 1 CPU

- RAM (sufficient for HP-UX and the applications in that partition)
- a boot disk (when using a mass storage unit, please check your hardware manual to verify that it can support a boot disk)

Although not required for booting a virtual partition, you can add LAN card(s) as required for networking.

For your virtual partitions, use the number of CPUs, amount of memory, boot disk configuration, and lan cards as is appropriate for your OS and applications.

CPUs

For detailed information on CPU allocation, please read “CPU” on page 179.

The `ioscan` output for the example non-cellular and cellular systems show the following processors:

```
keira# ioscan -kC procesor
H/W Path      Class                        Description
======================================================
0/10          processor                   Processor
0/11          processor                   Processor
0/12          processor                   Processor
0/13          processor                   Processor
0/14          processor                   Processor
0/15          processor                   Processor
1/10          processor                   Processor
1/11          processor                   Processor
1/12          processor                   Processor
1/13          processor                   Processor
1/14          processor                   Processor
1/15          processor                   Processor


winona# ioscan -kC processor
H/W Path      Class                         Description
======================================================
33            processor                    Processor
37            processor                    Processor
41            processor                    Processor
45            processor                    Processor
97            processor                    Processor
101           processor                    Processor
105           processor                    Processor
109           processor                    Processor
```

vPars A.04.xx and later

For this example, `keira1` will have two CPUs, `keira2` will have two CPUs, and `keira3` will have one CPU.

Partition Name	keira1	keira2	keira3

Assigned CPUs	`num = 2`	`num = 1 and`
		`1 from cell 1`
			`num = 1`

We have three CPUs that were not assigned to any of the virtual partitions, so we will have three CPUs available.

Unassigned CPUs	`three CPUs are available`

vPars A.03.xx and earlier

For this example, `winona1` will have two bound CPUs, `winona2` will have two bound CPUs where the hardware paths will be `41` and `45`, and `winona3` will have one bound CPU.

Partition Name	winona1	winona2	winona3

Bound CPUs	`total = 2`
	`min = 2`
		`total = 2`
		`min = 2`
		`paths = 41,45`
			`total = 1`
			`min = 1`

Unbound CPUs are assigned in quantity. We have three CPUs that were not assigned to any of the virtual partitions, so we will have three unbound CPUs available.

Unbound CPUs	`three CPUs are available`

Memory

For detailed information on memory allocation, please read “Memory: Allocation Concepts and Notes” on page 178.

In our examples, we will use the following sizes:

Partition Name	keira1	keira2	keira3
Memory	`1024 MB`	`1024 MB`	`1024 MB`

Partition Name	winona1	winona2	winona3
Memory	`1024 MB`	`1280 MB`	`1280 MB`

IO

For detailed information on IO Assignments, see “IO: Allocation Notes” on page 162.

For simplified IO block diagrams of the LBA to physical slot relationship of PA-RISC systems, see Appendix A, “LBA Hardware Path -> Physical IO Slot Correspondence (PA-RISC only),” on page 261.

Assigning IO at the LBA Level

Looking at the full ioscan output to verify that we have the desired IO for each virtual partition, we will assign the IO at the LBA level. (When assigning hardware at the LBA level to a partition, all hardware at and below the specified LBA is assigned to the partition.):

For our example servers, the ioscan output shows the LBAs as:

```
keira#ioscan -kC lba
0/0/0              ba                                  Local PCI Bus Adapter (782)
0/0/1              ba                                  Local PCI-X Bus Adapter (783)
0/0/2              ba                                  Local PCI-X Bus Adapter (783)
0/0/4              ba                                  Local PCI-X Bus Adapter (783)
0/0/6              ba                                  Local PCI-X Bus Adapter (783)
0/0/8              ba                                  Local PCI-X Bus Adapter (783)
0/0/10             ba                                  Local PCI-X Bus Adapter (783)
0/0/12             ba                                  Local PCI-X Bus Adapter (783)
0/0/14             ba                                  Local PCI-X Bus Adapter (783)
1/0/0              ba                                  Local PCI Bus Adapter (782)
1/0/1              ba                                  Local PCI-X Bus Adapter (783)
1/0/2              ba                                  Local PCI-X Bus Adapter (783)
1/0/4              ba                                  Local PCI-X Bus Adapter (783)
1/0/6              ba                                  Local PCI-X Bus Adapter (783)
1/0/8              ba                                  Local PCI-X Bus Adapter (783)
1/0/10             ba                                  Local PCI-X Bus Adapter (783)
1/0/12             ba                                  Local PCI-X Bus Adapter (783)
1/0/14             ba                                  Local PCI-X Bus Adapter (783)
```

```
winona#ioscan -k | grep "Bus Adapter"
H/W Path          Class        Description
============================================================
0/0               ba           Local PCI Bus Adapter (782)
0/1               ba           Local PCI Bus Adapter (782)
0/2               ba           Local PCI Bus Adapter (782)
0/4               ba           Local PCI Bus Adapter (782)
0/5               ba           Local PCI Bus Adapter (782)
0/8               ba           Local PCI Bus Adapter (782)
0/10              ba           Local PCI Bus Adapter (782)
0/12              ba           Local PCI Bus Adapter (782)
1/0               ba           Local PCI Bus Adapter (782)
1/2               ba           Local PCI Bus Adapter (782)
1/4               ba           Local PCI Bus Adapter (782)
1/8               ba           Local PCI Bus Adapter (782)
1/10              ba           Local PCI Bus Adapter (782)
1/12              ba           Local PCI Bus Adapter (782)
```

Partition Name	keira1	keira2	keira3

IO LBAs	`1.0.0 (boot)`
	`0.0.1 (lan)`
		`1.0.4 (boot)`
		`1.0.1 (lan)`
			`0.0.0 (boot)`
			`0.0.2 (lan)`

Partition Name	winona1	winona2	winona3

IO LBAs	`0.0 boot/lan`
	`0.4`
		`0.8 boot`
		`1.10 lan`
			`0.5 lan`
			`1.4 boot`

Ensuring the Hardware Console Port Is Owned by the First Virtual Partition (PA-RISC)

In our example server, the hardware console port is at `0/0/4/0`, which uses the LBA at `0/0`. The LBA `0/0` is owned by the partition `winona1`:

console port	`0.0.4.0`	`1/0/0/0/1`
LBA	`0.0`	`1.0.0`
partition	`winona1`	`keira1`
console port	`owned by winona1`	`owned by keira1`

When we create the virtual partitions, we will create `winona1` and `keira1` first.

CAUTION One of the virtual partitions must own the LBA that contains the physical hardware console port.

Choosing the Boot and Lan Paths

Using the full `ioscan` output, we chose the following boot disk path and note the LAN card path:

Partition Name	keira1	keira2	keira3
Boot Path	`1/0/0/3/0.6.0.0.6.0`	`1/0/4/1/0/4/0.1.0.0.0.0.1`	`0/0/0/3/0.6.0`
LAN	`0/0/1/1/0/4/0`	`1/0/1/1/0/4/0`	`0/0/2/1/0`

Partition Name	winona1	winona2	winona3
Boot Path	`0.0.2.0.6.0`	`0.8.0.0.5.0`	`1.4.0.0.5.0`
LAN	`0.0.0.0`	`1.10.0.0.4.0`	`0.5.0.0.4.0`

Autoboot

Autoboot allows a virtual partition to be booted automatically on a cold boot of the system. By default, autoboot is set to `AUTO` for all virtual partitions.

Partition Name	keira1	keira2	keira3
Autoboot	`AUTO`	`MANUAL`	`AUTO`

Partition Name	winona1	winona2	winona3
Autoboot	`AUTO`	`AUTO`	`AUTO`

For more information, see the *vparmodify* (1M) manpage.

NOTE When using `vparboot -I` to install vPars, you need to leave the autoboot attribute set to `AUTO` *during the installation* due to the reboots that occur during the installation. After installation is complete, you can set the autoboot attribute to `MANUAL` using the `vparmodify` command. For example, after installation is complete, to set the autoboot attribute to `MANUAL` for the partition `winona3`:

```
# vparmodify -p winona3 -B manual
```

Virtual Partition Plan

Combining all parts above, the resultant partition plans are the following:

Partition Name	keira1	keira2	keira3

Assigned CPUs (A.04.xx)	`num = 2`	`num = 1 and`
		`1 from cell 1`
			`num = 1`

Unassigned CPUs (A.04.xx)	`three CPUs are available`		
Memory	`1024 MB`	`1024 MB`	`1024 MB`

IO LBAs	`1.0.0`
	`0.0.1`
		`1.0.4`
		`1.0.1`
			`0.0.0`
			`0.0.2`

Boot Path	`1/0/0/3/0.6.0.0.6.0`	`1/0/4/1/0/4/0.1.0.0.0.0.1`	`0/0/0/3/0.6.0`
LAN	`0/0/1/1/0/4/0`	`1/0/1/1/0/4/0`	`0/0/2/1/0`
console port (PA-RISC Only)	`owned by keira1`		
Autoboot	`AUTO`	`MANUAL`	`AUTO`

Partition Name	winona1	winona2	winona3

Bound CPUs (A.03.xx)	`total = 2`
	`min   = 2`
		`total = 2`
		`min   = 2`
		`paths = 41,45`
			`total = 1`
			`min   = 1`

Unbound CPUs (A.03.xx)	`three CPUs are available`		
Memory	`1024 MB`	`1280 MB`	`1280 MB`

IO LBAs	`0.0`
	`0.4`
		`0.8`
		`1.10`
			`0.5`
			`1.4`

Boot Path	`0.0.2.0.6.0`	`0.8.0.0.5.0`	`1.4.0.0.5.0`
LAN	`0.0.0.0`	`1.10.0.0.4.0`	`0.5.0.0.4.0`

console port (PA-RISC Only)	owned by winona1		
Autoboot	AUTO	AUTO	AUTO

New Hardware Path Formats (A.02.02)

Beginning with vPars version A.02.02, the way to specify hardware paths has changed. This was done so that older vPars configuration databases remain compatible with additional hardware that is being supported.

For example, given a path where its sequential digits are `4 0 1 0 0 0 0`, it is not possible to determine whether this path means a device at "`4/0/1/0/0.0.0`" or a device at "`4/0/1/0/0/0/0.0.0.0.0.0.0`". The former structure is `cell/sba/lba/dev/function`, and the latter structure is `cell/sba/lba/pci_bridge/dev/function` where `pci_bridge` has the format `m/n`. Therefore, the following rules have been created. These rules apply when using either the vPars GUI or the command-line interface.

When entering a hardware path, the sequence and number of slashes (`/`)
and dots (`.`) in the hardware path that you input determines the resultant hardware path as follows:

Table 3-1 Hardware Path Format Rules

Path Format Description	Example	What the vPars Commands will do with the Entered Path
one or more occurrences of the `/` and `.`	`0/1.`	path will be padded

only occurrences of the `/`	`1/0`
	`1/0/1`
		path will not be padded

only occurrences of the `.`	`1.0`
	`1.0.`	

In the above table, padding means to pad using `.0` up to six elements after the first dot.

Example Using Legacy Paths

If a path was entered using slashes and dots and while using pre-A.02.02 software, you cannot enter the same path format using A.02.02 or later software. You must enter the path using the exact same digits but with dots instead of slashes as delimiters.

For example, if a path using A.02.01 software was entered as:

```
# vparmodify -p winona2 -m io:4/0/1/0/0.0.0:BOOT
```

then `vparstatus` would show the path as "`4.0.1.0.0.0.0`". To do the same `vparmodify` command above but using A.02.02 or later, the command would be:

```
# vparmodify -p winona2 -m io:4.0.1.0.0.0.0:BOOT
```

To change the above path to be the `ALTBOOT` setting, the command is:

```
# vparmodify -p winona2 -m io:4.0.1.0.0.0.0:ALTBOOT
```

Example Using Combo Cards

Beginning with vPars A.02.02, when setting a path, you can either use one or more occurrences of the `/` and `.` in the path so that the resultant path is the correctly padded path (this path is the same as the path shown in the `ioscan` output), OR use a correctly padded path directly using only dots (this path is the same as the path shown in the `vparstatus` output).

In the former case above, the `ioscan` output for a combo-card (combination of SCSI and LAN on a single PCI card) may show:

```
disk 0 12/0/8/0/0/4/0.0.0      ...  SEAGATE ST39103LC
```

Then, the vPars command would be

```
# vparmodify -p winona2 -a io:12/0/8/0/0/4/0.0.0
```

Note that this path of `12/0/8/0/0/4/0.0.0` becomes correctly padded to `12.0.8.0.0.4.0.0.0.0.0.0.0`, in accordance to the table above. Subsequently, the `vparstatus` output would show this path as `12.0.8.0.0.4.0.0.0.0.0.0.0`, which can be used if you wish to cut and paste the path as in the command "`vparmodify -p winona2 -m io:12.0.8.0.0.4.0.0.0.0.0.0.0:BOOT`".

In the latter case above, the `vparstatus` output for a combo-card may be:

```
12:0.8.0.0.4.0.0.0.0.0.0.0 BOOT
```

Then, the vPars command would be:

```
# vparmodify -p winona2 \
-m io:12.0.8.0.0.4.0.0.0.0.0.0.0:ALTBOOT
```

4 Installing, Updating, or Removing vPars and Upgrading Servers with vPars

This chapter covers

- Ignite-UX
- Installing vPars
- Updating vPars to the Latest Version of vPars
- Upgrading Servers with vPars
- Removing vPars

CAUTION Hardware Paths on the vPars Command Line

— **Hardware Path Differences Between Cellular (nPartitionable) and Non-cellular Systems**

The hardware paths for some example system are formatted for non-cellular systems. For cellular systems, their hardware paths contain the prefix of the cell number. Therefore, on non-cellular systems, the path `0/0` refers to a `SBA/LBA` format. However, on cellular systems, the path `0/0` refers to a `cell/SBA` format. Please read the section "Planning, Installing, and Using vPars with an nPartitionable Server" on page 48 if you are using a cellular system.

— **Path Formats on the vPars Command Line**

For vPars A.03.01 or earlier, you must explicitly specify the `LBA` for IO allocation. Thus, for cellular systems on A.03.01 or earlier, you must use the `cell/SBA/LBA` format on the command line. If you use only the `cell/SBA` format the vPars commands *will not* assume that all `LBA`s under the specified `SBA` are to be included in the allocation. Doing so may cause the system to panic.

For vPars A.03.02 or later, you can use either the `cell/SBA` or `cell/SBA/LBA` format on the command line. The vPars commands *will* assume the command applies to all `LBA`s under the specified `SBA`.

Bundle Names

You can install vPars on an existing HP-UX installation directly from a depot, DVD, or by using an Ignite-UX server.

vPars Product Bundles

The vPars bundle names are:

Bundle Name	Description
T1335BC	vPars A.04.xx and later for HP-UX 11i v2
Feature11i	Required vPars enablement patches for vPars A.04.xx and later. This bundle can be obtained from the 11i v2 May 2005 Update OE DVD. When the Feature11i and T1335BC bundle are in the same depot, this bundle should be automatically selected when the bundle T1335BC is selected.
T1335AC	vPars A.03.xx and earlier for HP-UX 11i v1

CAUTION Within a vPars environment, all OSs and vPars version must be the same. In other words, vpar1, vpar2, ... vparN must all be running the same OS and vPars software. Mixing HP-UX 11.11 and 11.23 or vPars A.04.xx and A.03.xx is not allowed.

vPars-related Bundles (A.03.xx and earlier)

Products related to this release of vPars are:

Bundle Name	Description
B6826AA	Partition Manager for nPartitions (parmgr) (Required for vPars A.03.xx and earlier)
VPARMGR	vPars GUI (vparmgr) for vPars A.03.xx and earlier (Optional)

For information on product versions required to run with vPars, please see the document *HP-UX Virtual Partitions Ordering and Configuration Guide* available at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions.

Installing and Removing vPars-related Bundles

B6826AA (parmgr) The Partition Manager (parmgr) is required for installation of the vPars A.03.xx and earlier. This is true on both nPartitionable servers as well as non-nPartitionable servers (rp7400/N4000 and rp5470/L3000). It is normal to have this product installed on non-nPartitionable servers.

The Partition Manager (PARMGR) is available at http://www.hp.com/go/softwaredepot.

To install the Partition Manager bundle using the vPars CD:

```
# swinstall -s /cdrom B6826AA
```

To remove the Partition Manager product:

```
# /usr/sbin/swremove PartitionManager
```

Note that the PartitionManager product can be removed only *after* the vPars product is removed from a virtual partition.

VPARMGR (vPars A.03.xx and earlier) The vPars GUI (vparmgr) is not automatically installed when vPars is installed. If you wish to use vparmgr, you need to manually select this bundle during your Software Distributor (SD) or Ignite-UX session or add this bundle to your `swinstall` command line.

VPARMGR is available on the vPars CD and from the vPars web page at http://www.hp.com/go/softwaredepot.

To install the VPARMGR bundle using the vPars CD:

```
# swinstall -s /cdrom VPARMGR
```

To remove the VPARMGR product:

```
# /usr/sbin/swremove vParManager
```

NOTE The product VPARMGR is optional.

Installing vPars and the vPars-related Bundles from CD Below is an example of using the `swinstall` command line to install vPars from the CD:

```
# swinstall -s /cdrom -x autoreboot=true T1335AC
```

To install vPars and the vPars-related bundles:

```
# swinstall -s /cdrom -x autoreboot=true T1335AC VPARMGR B6826AA
```

Patches

See the *HP-UX Virtual Partitions Ordering and Configuration Guide* for information on the required patches for vPars.

Setting Up the Ignite-UX Server

If you are having problems with terminal emulation, see also “Ignite-UX and other Curses Applications” on page 25.

For complete information on Ignite-UX, see the document *Ignite-UX Administration Guide*.

Ignite-UX Versions

vPars A.03.xx and A.04.xx have different version requirements. This version information has been moved to e the *HP-UX Virtual Partitions Ordering and Configuration Guide*.

NOTE PA-RISC only: When using **Ignite-UX version C.06.xx or later,** the bootable kernel path has changed from

`/opt/ignite/boot/WINSTALL` in Ignite-UX B.05.xx and earlier

to

`/opt/ignite/boot/Rel_B.11.NN/WINSTALL` in Ignite-UX C.06.xx and later.

Thus, when using Ignite-UX C.06.xx or later, you must specify the absolute path for the bootable kernel for the `vparboot -I` command line. For more information and an example, see “(PA-RISC only) The WINSTALL Boot Kernel Paths with Different Versions of Ignite-UX and the vparboot -I command” on page 24.

Determining the Ignite-UX Version

To determine which version of Ignite-UX you are running, execute the command similar to:

```
# swlist -l fileset -a revision Ignite-UX.FILE-SRV-11-11
```

Example

If your `swlist` output shows

```
# Initializing...
# Contacting target "rust"...
#
# Target:  rust:/
#
Ignite-UX.FILE-SRV-11-11                  B.5.4.50
```

then your Ignite-UX version is B.5.4.50.

ITRC Ignite-UX Cookbook

For quick steps in setting up your Ignite-UX Server to include the vPars bundles, you can use the "cookbook" information provided by the ITRC. You can access this information by:

1. Go to the ITRC web site:

   `http://itrc.hp.com`

2. Under `maintenance and support`, select `search technical knowledgebase`

3. Enter the keyword `vPars`
4. You will see *Ignite-UX & vPars Cookbook*

Ignite-UX, the LAN, the LAN card, and `vparboot -I`

NOTE **Using `vparboot -p target_partition -I`**

On both PA-RISC and Integrity, *before* booting a virtual partition for installation (in other words, using `vparboot -p target_partition -I...`), please be sure that you have specified a boot disk using the `BOOT` attribute (`io:`*`boot_device`*`:BOOT`) for the virtual partition. This is performed during either the initial vparcreate or subsequent vparmodify commands when configuring the target virtual partition.

If you have not specified a boot disk, you will not see the `BOOT` attribute in the `vparstatus -v` output for the target virtual partition:

```
[IO Details]
   0.1
   0.8
   0.3
```

If you have, you will see the `BOOT` attribute in the `vparstatus -v` output:

```
[IO Details]
   0.1
   0.8
   0.3
   0.8.0.0.8.0.110.0.0.0  BOOT
```

PA-RISC:

The following is the sequence of events when a `vparboot -I` is issued from an existing virtual partition to boot a target virtual partition:

1. the virtual partition from which the `vparboot` command is run uses `tftp` to obtain WINSTALL and WINSTALLFS. Note that the network interface card of the target virtual partition (the virtual partition you are attempting to boot) is not used in this step. Neither is `bootp` used.
2. WINSTALL and WINSTALLFS are transferred to the vPars Monitor.
3. the Monitor places them into the memory of the target virtual partition.
4. the target virtual partition uses WINSTALL and WINSTALLFS to boot and contacts the Ignite-UX server for the remainder of the installation.

Therefore, you should ensure the following:

- the network interface card that is owned by the virtual partition from which the `vparboot` command is issued allows `tftp` between the Ignite-UX server and this network interface.

  You can check the tftp connection by verifying the following works:

  ```
  vpar1# tftp <ignite-ux_server>
  tftp> get /opt/ignite/boot/WINSTALL
  Received 20495138 bytes in 9.9 seconds
  ```

- the network interface card of the target virtual partition is able to connect to the Ignite-UX server. This is performed by either:
  - — being on the same subnet as the Ignite-UX server
  - — entering IP and route information on the Ignite-UX screen that will be displayed on the console during boot
  - — contacting a DHCP server (boot helper) to obtain the IP and route information to the Ignite-UX server

Note that only the vPars shell command `vparboot` can be used to boot subsequent virtual partition for installation (or recovery); the vPars Monitor command `vparload` cannot do this. Thus, you need at least one virtual partition successfully booted to use the `vparboot` command.

Integrity:

The following is the sequence of events when a `vparboot -I` is issued from an existing virtual partition to boot a target virtual partition for an Integrity System:

1. the vPars Monitor sets the necessary variables such that the EFI shell will execute a `lanboot select` from the target virtual partition.
2. `lanboot select` lists the lan cards that are supported for boot that are within the target virtual partition and prompts the user to select one of the listed cards.

   NOTE: you will need to switch to the console of the target virtual partition using `CTRL-A` to see the list of cards.
3. using the selected card, the target virtual partition performs a network boot and connects to the Ignite-UX server.
4. once connected, the target virtual partition uses tftp to download the bootable kernel and filesystem IINSTALL and IINSTALLFS
5. then, the target virtual partition uses IINSTALL and IINSTALLFS to boot and contacts the Ignite-UX server for the remainder of the installation.

Therefore, you should ensure the following:

- the target virtual partition owns a network card that is supported for boot on Integrity.

  For more information on supported network cards, see the section titled “Networking Cards” in the *HP-UX Virtual Partitions Ordering and Configuration Guide*.
- the selected network interface card (NIC) of the target virtual partition is able to connect to the Ignite-UX server. This is performed by either:
  - — being on the same subnet as the Ignite-UX server. Note that in this case, the MAC address of the selected NIC is in the Ignite-UX server’s `/etc/bootptab`.
  - — entering IP and route information on the Ignite-UX screen that will be displayed on the console during boot
  - — contacting a DHCP server (boot helper) to obtain the IP and route information to the Ignite-UX server. Note that in this case, the MAC address of the selected NIC must be in the boot helper’s `/etc/bootptab`.

CAUTION `lanboot select` connects to the first Ignite-UX server from which its gets a response. Make sure that the NICs MAC address is registered with only one Ignite-UX server or boot helper in the subnet. If there are more than one Ignite-UX servers in the subnet and if one of them does not contain the latest Ignite-UX software, booting from an incompatible kernel may bring down the entire nPartition.

Like on PA-RISC, only the vPars shell command `vparboot` can be used to boot subsequent virtual partition for installation (or recovery); the vPars Monitor command `vparload` cannot do this. Thus, you need at least one virtual partition successfully booted to use the `vparboot` command.

Other Considerations

This section covers:

- “Notes on Installing Server Firmware” on page 68
- “Setting the GSP Terminal Type” on page 69
- “Increase in Size of /stand File System” on page 70
- “VxFS (Veritas File System) (vPars A.03.xx)” on page 70

Notes on Installing Server Firmware

Installing Firmware for the systems running vPars must be done in a standalone (PA-RISC) or nPars (Integrity) mode. Once in standalone or nPars mode, the procedure for installing firmware on a system with vPars installed is the same as a system without vPars installed. Additional information is shown below. For information on specific firmware versions for your servers, see the *HP-UX Virtual Partitions Ordering and Configuration Guide*.

- **Non-nPartitionable Systems**

  On the rp5470/L3000 and rp7400/N4000 servers, for firmware patches to take effect in a vPars environment, follow this procedure:

  1. Shut down all the virtual partitions.
  2. Reboot the server into *standalone* mode using the primary path. This consists of the following:
     a. At the MON> prompt, type `reboot`
     b. If needed, interrupt the boot sequence at the BCH>, and using the primary path, boot `/stand/vmunix` instead of `/stand/vpmon`. For example:

```
BCH> bo pri
interact with IPL? y
.
.
.
ISL> hpux /stand/vmunix
```

NOTE The server must be in standalone mode for the patches to take effect, so please do not skip this step.

  3. Install the firmware patch as you would in a non-vPars environment. The firmware patch will reboot your server.
  4. After the firmware installation has completed, you can boot the Monitor and virtual partitions as you normally would.

     For example, if you have not modified your `AUTO` file in the LIF area to boot the vPars Monitor and virtual partitions, boot the vPars Monitor (for example, `ISL> hpux /stand/vpmon`) and then the virtual partitions (for example, `MON> vparload -auto`).

- **Mid-range Servers**

  Once in standalone or nPars mode, install the server firmware as you normally do.

- **Superdomes (PA-RISC and Integrity)**

  Upgrading firmware on a Superdome must be performed by Hewlett-Packard qualified service personnel only. Please contact your local HP Support Representative to schedule a convenient time for the firmware upgrade service.

Setting the GSP Terminal Type

Note: this section applies only to the rp5470/L3000 and rp7400/N4000 servers. You can skip this section for nPartitionable servers.

The Guardian Service Processor (GSP) provides multiple access methods for the console: the hardware console port, the remote-modem port, and the LAN console port. To avoid mismatches in terminal emulation which can cause strange results on your display, it is important to match the display type as set in the GSP to the display type of the terminal or terminal emulator that you are using. For example:

- If you are using a hardwired HP terminal or a LAN-based terminal emulator of type “hpterm”, set the GSP terminal-type setting to `hpterm`.
- If you are using a LAN-based terminal emulator of type “dtterm” or “xterm”, set the GSP terminal-type setting to `vt100`.

How to Set the GSP Terminal Type

Step 1. Access the GSP through the lan console, the remote-modem port, or a physically connected terminal.

Step 2. Use the `CA` command at the GSP prompt to modify the console attributes:

```
GSP> ca
```

Step 3. Answer “y” (yes) to indicate that you want to change the console port settings:

```
Do you want to modify the Local Console Serial Port settings? (Y/[N])  y
```

Step 4. Answer “n” (no) to the questions about modifying the “Serial Port bit rate” and the “Current Flow Control”

```
Current Local Console Serial Port bit rate:  9600 bits/s
Do you want to modify it? (Y/[N])  n
Current Flow Control:  Software
Do you want to modify it? (Y/[N])  n
```

Step 5. Indicate which terminal type you want to use, then answer “y” (yes) to confirm your change:

```
Enter Terminal Type ([Vt100] / Hpterm):
New Terminal Type: hpterm
Confirm? (Y/[N]): y
  -> Terminal Type will be updated.
```

Step 6. Answer n (no) to the question about updating the “Remote Console Serial Port Modem settings”:

```
Do you want to modify the Remote Console Serial Port Modem settings? (Y/[N]) n
```

You will see a message indicating the command execution will take a few seconds and then a message indicating that your settings have been updated.

The virtual partitions that you create will use this terminal-type setting for their virtual console displays.

TIP If you get a garbled display, you can press `Ctrl-L` to refresh the display.

Increase in Size of `/stand` File System

Due to the vPars files that will exist in `/stand`, you should increase by 100 MB the size of the `/stand` file system that you normally create.

VxFS (Veritas File System) (vPars A.03.xx)

To avoid hangs on VxFS file systems, please install kernel patch PHKL_27121 or its successor on the operating systems of each virtual partition. This patch is available from the IT Resource Center web site at http://itrc.hp.com.

Related Information

For information on the installation of HP-UX, see the manual "HP-UX 11i Installation and Update Guide".

For information on `swinstall` and software depots, see the manual "Software Distributor Administration Guide for HP-UX".

For information on using the vPars commands, see the following sections in the chapter Monitor and Shell Commands:

- "Managing: Creating a Virtual Partition" on page 125.
- "Monitor: Booting the vPars Monitor" on page 103.
- "Boot||Shut: Booting a Virtual Partition" on page 132
- "Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)" on page 135.

For more information on booting and boot devices on PA-RISC systems, see also the paper titled *Booting, Installing, Recovery, and Sharing in a vPars Environment from DVD / CDROM / TAPE / Network* available at http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions.

Installing vPars Using Ignite-UX (PA-RISC)

1. **Boot your system using the Ignite-UX server. If your Ignite server's IP address is ww.xx.yy.zz:**

```
BCH> bo lan.ww.xx.yy.zz install
Interact with IPL: n
```

2. **Using the Ignite-UX server, install HP-UX, desired patches, the Quality Pack bundle, the vPars bundle, and the desired vPars-related bundles onto the disk that will be the boot disk of the first virtual partition.**

   NOTE: So that the `TERM` variable will always be set correctly, you should ensure that the first virtual partition owns the hardware console port. For more information, see "Ensuring the Hardware Console Port Is Owned by the First Virtual Partition (PA-RISC)" on page 55.

3. **Use `ioscan` to verify the hardware addresses in your virtual partition plan.**

```
# ioscan
```

4. **Create the virtual partitions using the information you prepared in the virtual partition plan.**

   For example, the non-nPartitionable system running A.03:

```
# vparcreate -p winona1 -a cpu::2 -a cpu:::2 -a mem::1024 -a io:0.0 -a io:0.4 -a
io:0/0/2/0.6.0:BOOT
# vparcreate -p winona2 -a cpu::2 -a cpu:::2 -a cpu:41 -a cpu:45 -a mem::1280 -a io:0.8
-a io:1.10 -a io:0/8/0/0.5.0:BOOT
# vparcreate -p winona3 -a cpu::1 -a cpu:::1 -a mem::1280 -a io:0.5 -a io:1.4 -a
io:1/4/0/0.5.0:BOOT
```

   Or for the nPartitionable system running A.04:

```
# vparcreate -p keira1 -a cpu::2 -a mem::1024 -a io:0.0.1 -a io:1.0.0 -a
io:1/0/0/3/0.6.0:BOOT
# vparcreate -p keira2 -a cpu::1 -a cell:1:cpu::1 -a mem::1024 -a io:1.0.1 -a io:1.0.4
-a io:1/0/4/1/0/4/0.1.0.0.0.0.1:BOOT
# vparcreate -p keira3 -a cpu::1 -a mem::1024 -a io:0.0.2 -a io:0.0.0 -a
io:0/0/0/3/0.6.0:BOOT
```

NOTE If you need to set your ILM or CLM granularity to values different from the defaults, you must do this using the first `vparcreate` command. For example, the first `vparcreate` command which creates the vPars database (/stand/vpmon) would be:

```
# vparcreate -p keira1 -g ILM:1024 -g CLM:1024 -a cpu::2 -a mem::1024 -a
io:0.0.1 -a io:1.0.0 -a io:1/0/0/3/0.6.0:BOOT
```

For more information on granularity values, see "Memory: Setting Granularity Values" on page 172.

5. **Reboot the system.**

```
# shutdown -r
```

6. **Interrupt the boot process as your system comes back up to reach the ISL prompt.**

   ```
   BCH> bo pri
   interact with IPL: y
   ```

7. **At the ISL prompt, boot the Monitor and the first virtual partition.**

   Example:

   ```
   ISL> hpux /stand/vpmon vparload -p winona1
   ```

8. **From the console of the running virtual partition (in this example, the running virtual partition is `winona1`), if the `TERM` environment variable is set to `unknown`, change the `TERM` environment variable to `hpterm`. For example, in the POSIX shell the command is:**

   ```
   # export TERM=hpterm
   ```

9. **Continuing on the console of the running virtual partition (winona1), perform the following for each remaining virtual partition:**

   a. boot the target virtual partition from the running virtual partition using `vparboot`.

      The syntax is:

      ```
      # vparboot -p <target_partition> -I <ignite_server>,<WINSTALL_path>
      ```

      For our example, if the target partition is `winona2`, we would execute the following command from `winona1`:

      ```
      # vparboot -p winona2 -I ww.xx.yy.zz,/opt/ignite/boot/Rel_B.11.11/WINSTALL
      ```

      You will see a message similar to the following:

      ```
      <MON> winona2 loaded
      ```

   b. press `Ctrl-A` until you see the console of the target partition. The console will display the Ignite-UX installation interface.

   c. enter the boot disk path, lan info, hostname, and IP of the target partition into the Ignite-UX interface and install HP-UX, desired patches, the Quality Pack bundle, the vPars bundle, and the desired vPars-related bundles. As a result of this process, the virtual partition will automatically reboot.

TIP If you get a garbled display, you can press `Ctrl-L` to refresh the display.

Your system should now be booted with all virtual partitions up.

Installing vPars Using Ignite-UX (Integrity)

NOTE Lan cards are used for boot during installation on Integrity systems.

Unlike vPars on PA-RISC, vPars on Integrity uses the lan card of the target virtual partition for `lanboot`. Please check that your lan card is supported for boot on 11.23 Integrity systems. For more information, see the *HP-UX Virtual Partitions Ordering and Configuration Guide*.

Also, the IO card and diagnostic utilities, such as `FCFUPDATE` and `IODIAG.efi`, will not operate in `vPars` mode; you must be in `nPars` mode. If you need to update firmware, it will be easier if you do so before booting in vPars mode. For information on modes, see “Modes: Switching between nPars and vPars Modes (Integrity only)” on page 95.

1. **Prepare the nPartition for Installation of HP-UX by setting the nPartition’s `apiconfig` setting to `default` (for install of HP-UX) and verifying the hardware path of the first virtual partition.**

   Example:

   a. apiconfig

   ```
   Shell> acpiconfig
   Acpiconfig settings: default
   ```

   (Note: if you must change the acpiconfig setting from windows to default, you must first enter the acpiconfig default command, and then immediately enter reset to reboot using the new setting.)

   b. hardware path

   ```
   fs0: Acpi(000222F0,915)/Pci(0|0)/Scsi(Pun6,Lun0)/HD(Part1,SigF4250000)
   ```

2. **From EFI, boot your system using the Ignite-UX server.**

   Example:

   ```
   Shell> lanboot select
      01  Acpi(000222F0,0)/Pci(1|0)/Mac(00306E0E5268)

   Select Desired LAN: 1
   Selected Acpi(000222F0,0)/Pci(1|0)/Mac(00306E0E5268)
   Running LoadFile()
   CLIENT MAC ADDR: 00 30 6e 0e 52 68
   DHCP…
   ```

NOTE `lanboot` information

Note that `lanboot` will show only the lan cards that are supported for boot with your existing configuration. If the card(s) you expect to see are not displayed, it may be necessary to issue a `reconnect -r` at the EFI prompt. Then try `lanboot select` again.

Example:

```
Shell> reconnect -r
Shell> map -r
```

3. **Using the Ignite-UX server, install the necessary bundles.**

   This includes HP-UX OE, any desired patches, the Quality Pack bundle, the vPars bundle, and any desired vPars-related bundles onto the disk that will be the boot disk of the first virtual partition.

4. **Use `ioscan` to verify the hardware addresses in your virtual partition plan. You can also save this output since once within a vPars environment, ioscan will only show the hardware that can be seen from the local partition.**

   ```
   # ioscan
   ```

5. **Create the virtual partitions using the information you prepared in the virtual partition plan.**

   Example:

   ```
   # vparcreate -p keira1 -a cpu::2 -a mem::1024 -a io:0.0.1 -a io:1.0.0 -a
   io:1/0/0/3/0.6.0:BOOT
   # vparcreate -p keira2 -a cpu::1 -a cell:1:cpu::1 -a mem::1024 -a io:1.0.1 -a io:1.0.4
   -a io:1/0/4/1/0/4/0.1.0.0.0.0.1:BOOT
   # vparcreate -p keira3 -a cpu::1 -a mem::1024 -a io:0.0.2 -a io:0.0.0 -a
   io:0/0/0/3/0.6.0:BOOT
   ```

NOTE If you need to set your ILM or CLM granularity to values different from the defaults, you must do this using the first `vparcreate` command. For example, the first `vparcreate` command which creates the vPars database (/stand/vpmon) would be:

```
# vparcreate -p keira1 -g ILM:1024:y -g CLM:1024:y -a cpu::2 -a mem::1024 -a
io:0.0.1 -a io:1.0.0 -a io:1/0/0/3/0.6.0:BOOT
```

For more information on granularity values, see “Memory: Setting Granularity Values” on page 172.

---

6. **Set the nPartition to boot into vPars mode.**

   ```
   # vparenv -m vPars
   ```

7. **Reboot the system and from the EFI Boot Manager, run the EFI shell.**

   ```
   # shutdown -ry 0
   ```

8. **From the EFI shell, boot the Monitor and the first virtual partition.**

   ```
   Shell> fs0:
   fs0:\> hpux
   HPUX> boot /stand/vpmon vparload -p keira1
   ```

9. **From the console of the running virtual partition, if the `TERM` environment variable is set to `unknown`, change the `TERM` environment variable to `hpterm`. For example, in the POSIX shell the command is:**

   ```
   keira1# export TERM=hpterm
   ```

10. **Continuing on the console of the running virtual partition (keira1), perform the following for each remaining virtual partition:**

    a. **boot the target virtual partition from the running virtual partition using `vparboot`.**

       The syntax is:

```
# vparboot -p <target_partition> -I
```

For our example, if the target partition is `keira2`, execute the following command from `keira1`:

```
# vparboot -p keira2 -I
```

You will see messages similar to the following:

```
<MON> keira2 loaded
```

b. **press `Ctrl-A` until you see the console of the target partition.**

```
[keira2]
```

c. **Select a MAC address from the list to perform a LAN boot.**

```
01  Acpi(000222F0,0)/Pci(1|0)/Mac(00306E0E5268)
Select Desired LAN: 1

Selected Acpi(000222F0,0)/Pci(1|0)/Mac(00306E0E5268)
Running LoadFile()
CLIENT MAC ADDR: 00 30 6e 0e 52 68
DHCP…
```

d. **Using the Ignite-UX server, install the necessary bundles.**

This includes HP-UX OE, any desired patches, the Quality Pack bundle, the vPars bundle, and any desired vPars-related bundles.

Your system should now be booted with all virtual partitions up.

Installing vPars Using Software Distributor (PA-RISC or Integrity)

1. For the root disk of each virtual partition, use Software Distributor to install HP-UX, desired patches, the Quality Pack bundle, the vPars software bundle, and the desired vPars-related bundles.
2. Boot the disk that is intended to be the boot disk of the first virtual partition into the normal (non-vPars) HP-UX environment.

   In our example, if the primary path is set to the boot disk of the first virtual partition `keira1`, the command is:

   - PA-RISC

     ```
     BCH> bo pri
     interact with IPL: n
     ```

     NOTE: So that the `TERM` variable will always be set correctly, you should ensure that the first virtual partition owns the hardware console port. For more information, see "Ensuring the Hardware Console Port Is Owned by the First Virtual Partition (PA-RISC)" on page 55.

   - Integrity

     ```
     Shell> fs0:
     fs0:\> hpux
     HPUX> boot vmunix
     ```

3. Use `ioscan` to verify the hardware addresses in your virtual partition plan:

   ```
   # ioscan
   ```

4. Create the virtual partitions using the information you prepared in the virtual partition plan.

   For example:

   ```
   # vparcreate -p keira1 -a cpu::2 -a mem::1024 -a io:0.0.1 -a io:1.0.0 -a
   io:1/0/0/3/0.6.0:BOOT
   # vparcreate -p keira2 -a cpu::1 -a cell:1:cpu::1 -a mem::1024 -a io:1.0.1 -a io:1.0.4
   -a io:1/0/4/1/0/4/0.1.0.0.0.0.1:BOOT
   # vparcreate -p keira3 -a cpu::1 -a mem::1024 -a io:0.0.2 -a io:0.0.0 -a
   io:0/0/0/3/0.6.0:BOOT
   ```

5. If you are on an Integrity system, set the mode to vPars; otherwise, you will not be able to boot into the vPars environment:

   ```
   # vparenv -m vPars
   ```

6. Reboot the system.

   ```
   # /etc/shutdown -r
   ```

7. Boot the system to ISL or EFI shell and boot the Monitor and all the virtual partitions:

   - PA-RISC

     ```
     BCH> bo pri
     interact with IPL: y
     ISL> hpux /stand/vpmon vparload -all
     ```

   - Integrity

```
Shell> fs0:
fs0:\> hpux
HPUX> boot /stand/vpmon vparload -all
```

Your system should now be booted with all virtual partitions up.

Updating from vPars A.04.xx to A.04.xx

This section covers how to update from an earlier version of vPars A.04.xx to the latest version of vPars A.04.xx.

NOTE The process documented here assumes you are not performing a hardware upgrade that causes a change in hardware paths (for example, upgrading from the sx1000 chipset to the sx2000 chipset). For information on upgrading vPars when the upgrade includes a hardware path change, please see “Upgrading Servers from the sx1000 to sx2000 Chipset (Integrity)” on page 86.

Step 1. Save a copy of your vPars database in case you need to revert back to the earlier version of vPars or you need to restore the database. The default location is /stand/vpdb.

Example:

```
keira1# cp /stand/vpdb /stand/vpdb.b4.update
```

Step 2. Make sure all the vPars are up and running. This allows you to update all the virtual partitions to the same vPars version; mixing vPars versions with A.04.01 and A.04.02 is not supported.

Example:

```
keira1 # vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name         State Attributes   Kernel Path           Opts
============================== ===== ============ ======================= =====
keira1                         Up    Dyn,Auto,Nsr /stand/vmunix
keira2                         Up    Dyn,Manl,Nsr /stand/vmunix
keira3                         Up    Dyn,Auto,Nsr /stand/vmunix
```

Step 3. Record the current autoboot and autosearch attributes of all the virtual partitions.

Example:

```
keira1 # vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name         State Attributes   Kernel Path           Opts
============================== ===== ============ ======================= =====
keira1                         Up    Dyn,Auto,Nsr /stand/vmunix
keira2                         Up    Dyn,Manl,Nsr /stand/vmunix
keira3                         Up    Dyn,Auto,Nsr /stand/vmunix
```

Step 4. Turn off the autoboot and autosearch attributes for all the virtual partitions.

Example:

```
keira1 # vparmodify -p keira1  -B manual -B nosearch
keira1 # vparmodify -p keira2  -B manual -B nosearch
keira1 # vparmodify -p keira3  -B manual -B nosearch
```

Step 5. On all the virtual partitions, `swinstall` the latest vPars bundle on each virtual partition.

Example:

```
keiraN# /usr/sbin/swinstall -x autoreboot=true depot1:/vpars/a.04.02 T1335BC
```

NOTE: if you are using alternate boot disks, you will need to boot the alternate boot disks and swinstall the latest vPars bundle to those boot disks as well.

Step 6. Once all the virtual partitions have shutdown as part of the swinstall reboot process, you will be at a vPars Monitor prompt. From here, you can reboot the nPartition.

Example:

```
MON> reboot
```

When the system comes back up to the MON> prompt, it should have loaded the new (latest) vPars Monitor.

Step 7. From here, reboot all the virtual partitions; they will complete their swinstall process.

Example:

```
MON> vparload -all
```

Step 8. Reset the autoboot and autosearch attributes of all the virtual partitions to their original values, which were recorded in Step 3.

Example:

```
keira1 # vparmodify -p keira1  -B auto -B nosearch
keira1 # vparmodify -p keira2  -B manual -B nosearch
keira1 # vparmodify -p keira3  -B auto -B nosearch
```

The virtual partitions should now be running the latest vPars A.04.xx version.

Updating from vPars A.03.xx to A.04.xx with Update-UX (PA-RISC only)

This section describes how to update an existing A.03.xx PA-RISC vPars environment to the latest A.04.xx PA-RISC vPars environment. The advantages of using this process are (1) you can update both OE and vPars versions simultaneously, so there are fewer reboots, and (2) although you must still reboot the nPartition, you can perform these steps within a vPars environment; you do not need to boot the system into standalone mode. However, if you are not familiar with Update-UX, you can continue to use the swinstall methods (see the links in the next paragraph).

Note that this process works only using Update-UX and a corresponding Ignite-UX depot; it does not work by directly using the OE and vPars media. If you wish to install directly from media, you should use the instructions from "Installing vPars Using Ignite-UX (PA-RISC)" on page 72 or "Installing vPars Using Software Distributor (PA-RISC or Integrity)" on page 77.

NOTE Using Update-UX

Update-UX allows for both the OE and vPars bundles to be updated in the same session. **Note that the OE and vPars bundles need to be in the same source depot.** See the ITRC Ignite-UX cookbook for information on how to setup your Ignite-UX server.

Before using the Update-UX command, make sure you have the latest Update-UX bundle installed for each virtual partition:

```
# swinstall -s <source_depot> Update-UX
```

In the example below, the Update-UX command syntax is

```
# update-ux -s <source_depot> <OE_bundle> <vPars_bundle>
```

For example, the command line used in this section is

```
# update-ux -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD HPUX11i-OE-Ent T1335BC
```

where

`depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD` is the source depot
`HPUX11i-OE-Ent` is the OE bundle
`T1335BC` is the vPars bundle

For more information on using Update-UX, please see the HP-UX 11.23 *Installation and Update Guide*.

The Steps

The following steps should be done **from the console**:

1. **Make sure that all the virtual partitions are up. You can check this with `vparstatus`.**

   Example:

```
keira1# vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name         State Attributes   Kernel Path            Opts
============================== ===== ============ ====================== =====
keira1                         Up    Dyn,Auto,Nsr /stand/vmunix
```

```
keira2                         Up    Dyn,Manl,Nsr /stand/vmunix
keira3                         Up    Dyn,Auto,Nsr /stand/vmunix
```

2. **Record the current autoboot and autosearch settings of all the virtual partitions so that you can change back to these settings later. To find the current settings, use `vparstatus`.**

   Example:

```
keira1 # vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name         State Attributes   Kernel Path            Opts
============================== ===== ============ ====================== =====
keira1                         Up    Dyn,Auto,Nsr /stand/vmunix
keira2                         Up    Dyn,Manl,Nsr /stand/vmunix
keira3                         Up    Dyn,Auto,Nsr /stand/vmunix
```

3. **Turn autoboot and autosearch settings off using `vparmodify` for all the virtual partitions.**

   Example:

```
keira1 # vparmodify -p keira1  -B manual
keira1 # vparmodify -p keira1  -B nosearch
keira1 # vparmodify -p keira2  -B manual
keira1 # vparmodify -p keira2  -B nosearch
keira1 # vparmodify -p keira3  -B manual
keira1 # vparmodify -p keira3  -B nosearch
```

   Note that the `-B nosearch` option is valid for only vPars A.03.02 and later. If you are using an earlier version of vPars, you can skip the `vparmodify ... -B nosearch` command.

4. **Install the latest Update-UX bundle onto each virtual partition (use Ctrl-A to switch between consoles).**

   Example:

```
keira1 # swinstall -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD Update-UX
keira2 # swinstall -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD Update-UX
keira3 # swinstall -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD Update-UX
```

5. **For each virtual partition, except the first virtual partition, use Update-UX to install the latest OE and vPars bundle. These updates can occur in parallel, although this is not required.**

   The first virtual partition is defined as the virtual partition that owns the boot disk from which the Monitor was booted; you can use the vparstatus -m and vparstatus -v commands to determine which virtual partition this is.

   Although you can update all the virtual partitions, including the first virtual partition, in parallel, by leaving the first virtual partition up until all the updates for the other virtual partitions are complete, it allows you to use the first virtual partition to verify the processing and status of the other virtual partitions.

   Example:

```
keira2 # update-ux -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD HPUX11i-OE-Ent T1335BC
keira3 # update-ux -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD HPUX11i-OE-Ent T1335BC
```

Bundle Names

- For HP-UX 11i v2, the possible **OE bundles** are listed below.

`HPUX11i-OE`	Foundation OE
`HPUX11i-OE-Ent`	Enterprise OE
`HPUX11i-OE-MC`	Mission Critical OE

  You should chose the same OE that your current virtual partition is running. Use the `swinstall` command to check which OE you are currently running:

  ```
  # swlist -l bundle | grep -i OE
   HPUX11i-OE-Ent        B.11.23.0505 HP-UX Enterprise Operating Environment
  ```

- `T1335BC` is the vPars A.04.xx bundle.

NOTE Be sure that both the OE and vPars bundles are specified on the update-ux command line.

6. **After the all updates for the above virtual partitions have completed, use Update-UX to install the latest OE and vPars bundle to the first virtual partition.**

   Example:

   ```
   keira1 # update-ux -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD HPUX11i-OE-Ent T1335BC
   ```

   Although you can do all the updates in parallel, you need to make sure that all of the other virtual partition updates have successfully performed the updating to the point of halting for reboot. In the next step, the entire nPartition will be rebooted; if the other virtual partitions are still in progress of updating, the OS instances may be in an unknown state.

NOTE If the BOOT and ALTBOOT disks are a mirrored pair, updating is not required on the ALTBOOT disk. Otherwise, after updating the OS on the primary boot path disk, boot the virtual partitions from the alternate path boot disk and repeat the update-ux procedure. For example, if the primary boot path disk is in keira1 and the alternate boot path disk is in keira2, boot the alternate disk

```
MON> vparload -p keira2 -B ALT
```

and repeat the update-ux procedure

```
keira2# update-ux -s depot1:/release/0505.1123/HPUX11i-OE-Ent.DVD HPUX11i-OE-Ent T1335BC
```

7. **Reboot the nPartition to ISL>**

   Example:

   ```
   MON> reboot
   ```

NOTE At this point, you need to reboot the nPartition from the MON> prompt, not just the virtual partition. By rebooting the nPartition, you can load the new vPars Monitor in the next step.

8. **From ISL> load the vPars Monitor.**

   Example:

```
ISL> hpux /stand/vpmon
```

9. **Boot the virtual partitions.**

   Example:

   ```
   MON> vparload -all
   ```

   When the virtual partitions are booted, they will continue and complete their update processes. After this is completed, you should arrive at the login: prompt for each virtual partition. Login as root and continue to the next step.

10. **Turn autoboot and autosearch settings back to their original settings that you recorded earlier above.**

    Example:

    ```
    keira1 # vparmodify -p keira1  -B auto -B nosearch
    keira1 # vparmodify -p keira2  -B manual -B nosearch
    keira1 # vparmodify -p keira3  -B auto -B nosearch
    ```

    The virtual partitions should now be running the latest OE and vPars version.

Updating from vPars A.02.xx or A.03.xx to A.03.xx (PA-RISC only)

To update from an earlier vPars A.02.xx or A.03.xx version to the latest vPars A.03.xx version, perform the following:

Step 1. Save a copy of the vPars database in case you need to revert back to the earlier version of vPars or you need to restore the database.

Step 2. Shut down all the virtual partitions.

Step 3. Reboot the server into standalone mode. This consists of the following:

1. At the MON> prompt, type `reboot`
2. If needed, interrupt the boot sequence at the BCH> and boot `/stand/vmunix` instead of `/stand/vpmon`. For example:

```
BCH> bo pri
interact with IPL? y
.
.
.
ISL> hpux /stand/vmunix
```

Step 4. Install the new vPars bundle using `swinstall`. For the appropriate bundle name, see "Bundle Names" on page 62.

Step 5. Reboot the system:

```
# shutdown -r
```

Step 6. Boot the vPars Monitor and the virtual partitions from the disk where you installed the vPars bundle. For example, if we had installed the bundle to the disk at the primary path:

```
BCH> bo pri
interact with IPL? y
.
.
.
ISL> hpux /stand/vpmon -a
```

If you have configured your `AUTO` file in the LIF area to boot the Monitor and virtual partitions automatically, then this step will be performed automatically. For more information, see "Boot||Shut: Autoboot" on page 143.

Step 7. On each virtual partition, repeat Step 4 to install the new vPars bundle on each boot disk of each virtual partition (you do not need to reboot the hard partition). Because the boot disk used to boot in standalone mode in Step 3 already has the new vPars bundle (this was installed during Step 4), you can exclude this step for the boot disk at the primary path.

You must install the new vPars bundle on each virtual partition before putting the virtual partitions back into production. Running a mix of older and newer vPars versions within a group of virtual partitions is not supported.

Upgrading Servers from the sx1000 to sx2000 Chipset (Integrity)

You can upgrade the following Integrity servers from the sx1000 to sx2000 chipsets:

- rx7620 to rx7640
- rx8620 to rx8640
- Integrity Superdome

For the upgrade process steps, please see the hardware upgrade documentation for your server. You can also view the *LVM/VxVM and vPars sx2000 Upgrade* document available at the web site below:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

CAUTION Upgrade Notes

- vPars A.04.02 or later is required for vPars systems running the sx2000 chipset.
- PHKL_34088 is also required for vPars systems running the sx2000 chipset.
- The physical hardware upgrade from the rx7620 to rx7640 requires a cold-install of the OS and therefore of vPars as well.
- Please be aware that instead of performing the upgrades for vPars, LVM etc., you also have the option of performing a cold-install on the post-upgrade hardware as if they were new systems with no previous installations.

Hardware Path Changes

The upgrade changes the hardware paths. For the new hardware paths as well as the hardware upgrade process, see the upgrade manual for your server. For reference, the changes are noted in the tables at the end of this section:

- "rx7620 to rx7640 Hardware Path Changes" on page 87
- "rx8620 to rx8640 Hardware Path Changes" on page 87
- "Integrity Superdome Hardware Path Changes (x=cell)" on page 88.

vPars Database Changes

For the hardware upgrade process, follow the process documented in the hardware upgrade manual. As part of the process, you will need to perform the following changes to the vPars database (vpdb) in `nPars` mode. For more information on `nPars` mode, see "Modes: Switching between nPars and vPars Modes (Integrity only)" on page 95.

- updating hardware paths that are assigned to any virtual partitions using `vparmodify`:
  Examples:
  - To add a new boot path:
    ```
    # vparmodify -p vpar_name -a io:new_path:boot
    ```
  - To delete an old boot path:
    ```
    # vparmodify -p vpar_name -d io:old_path:boot
    ```
  - To add a new LBA:
    ```
    # vparmodify -p vpar_name -a io:new_path
    ```

— To delete an old LBA:

```
# vparmodify -p vpar_name -d io:old_path
```

- updating the EFI to hardware path mappings using `vparefiutil`:
  Example:

— To delete the old entries and update with the new entries:

```
# vparefiutil -d
# vparefiutil -u
```

Hardware Path Tables

The tables below show the new hardware paths. For details on the new hardware paths and other hardware information, see the applicable hardware upgrade manuals for your server, available at http://docs.hp.com.

Table 4-1 rx7620 to rx7640 Hardware Path Changes

rx7620 Path	rx7640 Path	Description
1/0/1/1/0/1/1.6	1/0/1/1/0/4/1.6	Top right HDD using A6794A or AB290A
0/0/0/3/0.6	0/0/0/3/0.6	Bottom left HDD using MP/SCSI
0/0/0/3/0.5	0/0/1/1/0/4/1.5	Bottom right HDD using MP/SCSI or AB290A
1/0/0/3/1.x	1/0/0/3/1.x	Internal DVD/tape (x=2 for DVD, x=3 for DAT)
	0/0/0/3/1.2	Bottom slimline DVD, if present
1/0/1/1/0/4/0	1/0/1/1/0/6/0	Primary LAN
	1/0/1/1/0/6/1	Secondary LAN
0/0/8/1/0/4/0	0/0/1/1/0/6/0	Primary LAN
	0/0/1/1/0/6/1	Secondary LAN
1/0/1/1/0/1/0.x	1/0/1/1/0/4/0.x	external SCSI (A6794A/AB290A)
	1/0/1/1/0/4/1.x	external SCSI (AB290A), shared with internal HDD
0/0/8/1/0/1/0.x	0/0/1/1/0/4/0.x	external SCSI (A6794A/AB290A)
	0/0/1/1/0/4/1.x	external SCSI (AB290A), shared with internal HDD

Table 4-2 rx8620 to rx8640 Hardware Path Changes

rx8620 Path	rx8640 Path	Description
0/0/0/0/0	N/A	simplecom
0/0/0/0/1	[rootcell]/250/2	console UART
1/0/0/0/0	N/A	simplecom
1/0/0/0/1	[rootcell]/250/2	console UART

Table 4-3 Integrity Superdome Hardware Path Changes (*x*=*cell*)

Slot	sx1000 Path	sx2000 Path
0	x/0/0/1	x/0/0/1
1	x/0/1/1	x/0/1/1
2	x/0/2/1	x/0/2/1
3	x/0/3/1	x/0/4/1
4	x/0/4/1	x/0/5/1
5	x/0/6/1	x/0/6/1
6	x/0/14/1	x/0/14/1
7	x/0/12/1	x/0/13/1
8	x/0/11/1	x/0/12/1
9	x/0/10/1	x/0/10/1
10	x/0/9/1	x/0/9/1
11	x/0/8/1	x/0/8/1

Upgrading Backplanes from PCI to PCI-X

If you upgrade from the PCI to PCI-X backplane with the the following server upgrades:

- rp7410 to rp7420
- the rp8400 to rp8420
- Superdome

the hardware paths of the IO devices will change. The IO device paths are in the format

```
cell/sba/lba/device/function.target.lun
```

When changing from the PCI to PCI-X backplane, the **device** in the hardware paths will change from `0` to `1`. So, if a hardware path before the upgrade was

```
1/0/8/0/0.6.0
```

the new hardware path is

```
1/0/8/1/0.6.0
```

Not all IO hardware paths change after the PCI to PCI-X backplane upgrade. For details on the new paths for each slot, see the respective hardware upgrade manuals available at http://docs.hp.com.

For vPars and other changes, including the correct LVM and VxVM modifications on a vPars system, please see the *LVM and vPars IO Backplane Upgrade* document at:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

Please be aware that instead of performing the upgrades for LVM, vPars, etc., you also have the option of performing a cold-install on the upgraded rp8420 or rp7420 as if they were new systems with no previous installations.

Applying a vPars Sub-System Patch

The vPars sub-system patch includes the vPars Monitor, commands, and daemons. To apply a vPars patch to an existing version, perform the following:

1. Shut down all the virtual partitions.
2. Reboot the server into standalone mode. This consists of the following:
   a. At the MON> prompt, type `reboot`
   b. If needed, interrupt the boot sequence at the BCH> and boot `/stand/vmunix` instead of `/stand/vpmon`. For example:
   ```
   BCH> bo pri
   interact with IPL? y
   .
   .
   .
   ISL> hpux /stand/vmunix
   ```
3. Install the vPars sub-system patch using `swinstall`.
4. Reboot the system:
   ```
   # shutdown -r
   ```
5. Boot the vPars Monitor and the virtual partitions from the disk where you installed the vPars sub-system patch. For example, if we had installed the patch to the disk at the primary path:
   ```
   BCH> bo pri
   interact with IPL? y
   .
   .
   .
   ISL> hpux /stand/vpmon -a
   ```
   If you have configured your `AUTO` file in the LIF area to boot the Monitor and virtual partitions automatically, then this step will be performed automatically. For more information, see “Boot||Shut: Autoboot” on page 143.
6. On each virtual partition, repeat Step 3 to install the vPars sub-system patch on each boot disk of each virtual partition. No reboot of the virtual partition is required.

   Because the boot disk used to boot in standalone mode in Step 2 already has the new vPars patch (this was installed during Step 3), you can exclude this step for the boot disk at the primary path.

   You must install the vPars patch on each virtual partition before putting the virtual partitions back into production. Running a mix of vPars products and versions within a group of virtual partitions is not supported.

NOTE *If you have previously applied any patches specific to vPars, you will need to re-apply those patches to the vPars product.*

Updates Involving VPARSBASE (PA-RISC)

VPARSBASE (the free demo product for HP-UX 11i v1) is no longer available or supported.

You can update directly only from free product to newer free product or from purchased product to newer purchased product. You cannot update directly from free product to the purchased product.

If you wish to update from the free product to the purchased product, first you need to remove the free product VPARSBASE, and then install the purchased product T1335AC for HP-UX 11i v1.

To avoid having to recreate your vPars configuration, you can perform the following steps:

1. Backup the file `/stand/vpdb` (this file is the vPars partition database)
2. Remove the free product from all virtual partitions. See “Removing the vPars Product” on page 92.
3. Install the full product. See “Installing vPars Using Ignite-UX (PA-RISC)” on page 72 or “Installing vPars Using Software Distributor (PA-RISC or Integrity)” on page 77.

   However, while in standalone mode, at the point where you would normally create the virtual partitions using vparcreate, restore `/stand/vpdb`.

   Then continue with the installation as you normally would.

Removing the vPars Product

From a Single Virtual Partition

To remove the vPars product, execute the `swremove` command from the target virtual partition. For example, to remove the vPars product from the partition `winona3`:

```
winona3# /usr/sbin/swremove -x autoreboot=true VirtualPartition
```

The product will be removed, and the virtual partition will be shut down. Because the vPars-specific kernel modifications have been removed, the OS instance cannot be booted again as a virtual partition. However, the OS instance can be booted into standalone mode.

If you are at the console of `winona3`, use `Ctrl-A` to toggle to another virtual partition.

NOTE When the vPars product is removed, the contents of the `AUTO` file in the LIF area will be set to the default `hpux` (where `/stand/vmunix` is the default argument). This is true even if you have previously modified the `AUTO` file to contain `hpux /stand/vpmon`. This replacement occurs on the boot devices associated with the target virtual partition, including the devices listed in `/stand/bootconf` and the primary and alternate paths of the target virtual partition. See *bootconf* (4) for more information on the file `/stand/bootconf`.

CAUTION Remove the vPars product only at the product level (`VirtualPartition`). Do NOT remove the vPars product at the bundle level (T1335AC or VPARSBASE). Recommended kernel patches are included in the vPars bundle; if the bundle is removed, these kernel patches will also be removed. For more information on bundles and patches, see the "Patch Management Guide for HP-UX 11.x" at http://docs.hp.com.

From the Entire System

1. Remove the vPars product from each virtual partition one by one.
2. After you have removed vPars from the last virtual partition, you will be at the Monitor prompt. At this point, you can type REBOOT to reboot the system.

   ```
   MON> reboot
   ```

NOTE On Integrity systems, you will also need to change the mode from vPars to nPars in order to be able to boot in nPars (standalone) mode.

To uninstall vparmgr or other vPars-related bundles, see “Installing and Removing vPars-related Bundles” on page 62.

5 Monitor and Shell Commands

This chapter covers:

- Using Integrity systems
  - — Setting Modes
  - — EFI to Hardware Path Mappings
- Using the vPars Monitor
  - — Booting the Monitor
  - — Accessing the Monitor Prompt
  - — Using Monitor Commands
- Using the vPars Commands
  - — vPars Manpages
  - — vPars Commands Logging
  - — Obtaining Monitor and Hardware Resource Information
- Managing the Virtual Partitions
  - — Creating a Virtual Partition
  - — Booting a Virtual Partition
  - — Shutting Down or Rebooting a Virtual Partition
  - — Shutting Down or Rebooting the Hard Partition
  - — When to Shutdown All Virtual Partitions
  - — Removing a Virtual Partition
  - — Using Primary and Alternate Boot Paths
  - — Autobooting the Monitor and All Virtual Partitions
  - — Resetting a Hung Virtual Partition
  - — Booting a Virtual Partition Into Single-User Mode
  - — Using Other Boot Options
  - — Simulating the AUTO File on a Virtual Partition
  - — Modifying Attributes of a Virtual Partition
- Using an Alternate Partition Database File

Notes on Examples in this Chapter

Syntax of Example Commands

The example commands at the Unix shell level in the following section use the following syntax:

```
<HP-UX shell prompt><command>
```

where the shell prompt consists of the hostname of the current virtual partition and the hash sign (`#`). For example, if we log into `winona1` and run the `ls` command, the command is shown as:

```
winona1# ls
```

If we are logged into `winona1` and run the `vparboot` command with `winona3` as the target virtual partition, the command is shown as:

```
winona1# vparboot -p winona3
```

Example Server

Some examples in this section use the same non-cellular rp7400/N4000 configuration as in the installation chapter. See “full ioscan output of non-cellular system named winona” on page 44.

Note: unlike the rp7400/N4000, on a Superdome and other nPartition servers, the first element of the hardware path of the ioscan output is the cell number. For example, on the rp7400/N4000 the ioscan output shows:

```
0/0        ba              Local PCI Bus Adapter (782)
```

However, on a cellular (nPartitionable) systems, the first element of the hardware path is the cell number. So, if the cell number is `4`, the ioscan output shows:

```
4/0/0      ba              Local PCI Bus Adapter (782)
```

NOTE *LBA must be explicitly specified when using vPars A.03.01 or earlier*. Please read “Planning, Installing, and Using vPars with an nPartitionable Server” on page 48.

Modes: Switching between `nPars` and `vPars` Modes (Integrity only)

Modes

On an Integrity system, you will need to set the mode in order to boot into a specific mode. For vPars usage, there are only two modes:

- **`vPars`**

  sets the next nPartition boot to boot into the vPars environment. This allows you to boot the vPars Monitor and therefore the virtual partitions in the next nPartition boot. You still need to boot the vPars Monitor and the virtual partitions, but this mode allows you to do this.

- **`nPars`**

  sets the next nPartition boot to boot into the standalone environment. In this mode, you cannot boot the vPars Monitor and therefore the virtual partitions. However, you can boot any OE instance into standalone mode.

You can set the mode from the following levels using the corresponding commands:

Location	Command
HP-UX Shell	`vparenv`
MON>	`reboot`
EFI	`vparconfig`

Commands to Set the Mode

- **HP-UX Shell: `vparenv [-m` *`mode`*`]`**

  where

mode	has the value of either `vPars` or `nPars`

  sets the mode for the next nPartition reboot. Note that this may take a few minutes to process.

  Example:

  To set the nPartition into vPars mode so that the next nPartition boot allows you to boot the vPars Monitor and therefore the vPars environment:

  1. Set the mode

     ```
     # vparenv -m vPars
     ```

  2. Then, you manually reboot the nPartition:

     ```
     # shutdown -r
     ```

```
...
Shell> fs0:
fs0:\> hpux /stand/vpmon
...
MON>
```

- **Monitor: `reboot` [*mode*]**

  where

mode	has the value of either `vPars` or `nPars`

  reboots the nPartition into the *mode* mode. If any virtual partitions are up, this will cause them to be shutdown ungracefully.

- **EFI: `vparconfig [reboot` *mode*`]`**

  where

mode	has the value of either `vPars` or `nPars`

  sets the *mode* for the next nPartition reboot and then also reboots the nPartition.

  Note that `vparconfig` is **not a built-in EFI shell** command. You must go to the disk (for example, `fs0:`) to execute the `vparconfig` command.

  Examples:

  - To set the mode to `vPars` and then immediately reboot the nPartition into `vPars` mode:

    ```
    Shell> fs0:                    /* first goto the disk */
    fs0:\> vparconfig reboot vPars /* then you can execute vparconfig */
    ```

  - To set the mode to `nPars` and then immediately reboot the nPartition into `nPars` mode

    ```
    Shell> fs0:                    /* first goto the disk */
    fs0:\> vparconfig reboot nPars
    ```

- **EFI: `parconfig [`*mode*`[-n] ]`**

  where

mode	has the value of only `nPars`. parconfig does not allow you to set the mode to `vPars`
`-n`	means no interactive prompts

  NOTE: HP recommends using `vparconfig` instead of `parconfig` whenever possible; information on `parconfig` is provided here as additional information or when `vparconfig` is not present on the disk. `vparconfig` is installed only on the boot disks of the virtual partitions when vPars is installed. If the boot disks are removed or you switch boot disks, you may need to use `parconfig`.

  Example

  To set the nPartition into `nPars` mode and reboot the nPartition:

  1. First, set the mode:

     ```
     Shell> parconfig nPars -n
     ```

  2. Then, you can reboot the nPartition from either the EFI shell using the `reset` option:

     ```
     Shell> parconfig reset
     ```

Differences Between `vparconfig` and `parconfig`

Table 5-1 vparconfig versus parconfig

	vparconfig	parconfig
EFI shell:	`vparconfig` is not a built-in EFI shell command, so you must execute `vparconfig` from the disk.	`parconfig` is a built-in EFI shell command, so you can execute `parconfig` from the EFI shell.
syntax:	`vparconfig reboot` *`mode`*	`parconfig` *`mode`*
nPartition reboot:	`vparconfig` automatically reboots the nPartition after you set the mode.	`parconfig` does not automatically reboot the nPartition. You must manually reboot the nPartition.

Usage Scenarios

- If you are running HP-UX in `nPars` mode (standalone), use the following vPars command to switch to `vPars` mode:

  ```
  OS-Prompt> vparenv -m vPars       /* sets the mode for the next nPartition reboot */
  OS-Prompt> reboot                     /* to reboot the system into vPars mode */
  ```

- If you are at the Monitor prompt, use the following Monitor command to switch to `nPars` mode:

  ```
  MON> reboot nPars                 /* sets the mode and reboots the system */
  ```

- If you are at EFI shell prompt, use the following EFI utility to switch to either `nPars` or `vPars` mode:

  ```
  Shell:> fsN:
  fsN:> vparconfig reboot nPars|vPars
  ```

  Since `vparconfig` is not a built-in EFI shell command, you must goto the disk to execute `vparconfig`. For example, to switch to `vPars` mode:

  ```
  Shell:> fs0:                       /* goto the EFI partition of the disk */
  fs0:> vparconfig reboot vPars     /* sets the mode and reboots the system  */
  ```

  Note: `vparconfig` is an EFI utility which gets installed in the EFI partition during the installation of the vPars product.

- If you are at EFI shell prompt in `vPars` mode and you do not have vPars installed on any of your disks, you can use the built-in EFI command `parconfig` to switch to `nPars` mode:

  ```
  Shell:> parconfig nPars
  Shell:> parconfig reset
  ```

  Note: Remember to issue a `parconfig reset` after setting the mode. `parconfig nPars` only sets the mode to `nPars`. You must issue the `parconfig reset` to reset the system so that it boots into `nPars` mode.

  Note: `parconfig` does not support switching to `vPars` mode. In other words, you can use `parconfig` to set the mode to `nPars`, but you cannot use `parconfig` to set the mode to `vPars`.

- During a cold-install of the OE and vPars software, the following general steps could occur:

  1. Boot and install the OE and vPars software as well as create the vPars database onto the intended boot disk of a virtual partition.

2. Set the mode to `vPars` so that you can boot the nPartition into the vPars environment.

   ```
   # vparenv -m vPars
   ```

3. Reboot the nPartition into the vPars environment and load the first virtual partition.
4. From the first virtual partition, use vparboot -I to install the OE and vPars software onto the remaining boot disks of the remaining virtual partitions.

For detailed steps on how to do the installation, see "Installing vPars Using Ignite-UX (Integrity)" on page 74.

- Suppose you have booted to the Monitor prompt but are unable to load the any vPars databases. You can boot the system into `nPars` (standalone) mode and attempt to look into the database without the Monitor running. To do this:

  1. Set the mode to nPars and reboot the nPartition:

     ```
     MON> reboot nPars
     ```

  2. During the nPartition bootup process, boot into standalone mode by booting the `vmunix` kernel instead of the vPars Monitor:

     ```
     Shell> fs0:
     fs0:\> hpux.efi /stand/vmunix
     # vparstatus -v -D /stand/vpdb
     ```

CAUTION

- When you set the mode to `vPars` for the first time on a system, you must use `vparenv`.

  When a vPars database does not exist on a system, first boot into `nPars` (standalone) mode, create the vPars database, and then use `vparenv -m vPars` to switch the mode to `vPars`.

  If `vparconfig reboot vPars` is used and `vparenv -m vPars` has not previously been executed on the system, it may not be possible to boot vPars.

- Changing the mode to `vPars` should be performed using `vparenv` instead of `vparconfig` whenever possible.

NOTE

- When the system is at the EFI shell prompt in `vPars` mode, you can use either one of the following commands to reset the nPartition:

  — `EFI_Shell> parconfig reset`

  — ```
    EFI_Shell> fsx:
    fsx> vparconfig reboot vPars
    ```

  The standard EFI Shell command `reset` *should not* be used to reset the system or nPartition when it is in `vPars` mode.

- If the desired mode is not set, you will not be able to boot into that mode. For example, you will not be able to boot the vPars Monitor (/stand/vpmon) when you are in nPars mode. Likewise, you will not be able to boot into standalone mode when you are in vPars mode.
    ```

- On an Integrity system which has vPars software installed but does not have the correct firmware version installed, you will see the following behavior depending upon the mode of operation:
  - — If the current mode is `nPars`, booting `vmunix` works as expected. Booting `vpmon` exits with an `unsupported environment` message.
  - — If the current mode is changed to `vPars` using `vparenv` or `vparconfig`, the hpux loader does not allow boot of either the `vpmon` or `vmunix`. In this case, you should use `vparconfig` to change mode back to `nPars` and reboot the system. You should then install the required firmware. See the *HP-UX Virtual Partition Ordering and Configuration Guide* for information on the required firmware.

EFI Boot Disk Paths, including Disk Mirrors, and `vparefiutil` (Integrity only)

On PA-RISC systems, the bootloader can boot a disk using only the hardware path of the disk. However, on Integrity systems, the bootloader requires the EFI path. On Integrity systems running vPars, the vPars database contains the initial hardware path to EFI path mappings; on boot of a virtual partition, the vPars Monitor transparently provides the EFI path from the vPars database to the bootloader so that a virtual partition can boot.

The EFI path changes whenever the boot area changes on the disk. During the initial creation of the vPars database, during the installation of an OS using `vparboot -I`, and during the execution of the `setboot` command, the EFI paths are updated in the vPars database.

However, beyond the above situations, whenever the EFI path of an existing boot disk changes or an additional boot disk is added, including adding a boot disk mirror, the EFI mappings within the vPars database need to be updated. Otherwise, the virtual partition may not boot. Note that using `vparmodify` to change a boot path in the vPars database does not update the EFI path in the vPars database.

To update the EFI path of a boot disk in the vPars database (for example, after creating a boot disk mirror), execute **`vparefiutil -u`** (-u for update) on each virtual partition. Other examples of when you should use `vparefiutil` include:

- For a specific HP-UX hardware path, if there is no EFI path mapping, the `vparboot` and `vparload` commands will fail with the following error message:

  ```
  Primary boot path not found.
  Internal error in setting up vPars variables.
  "vpar" load failed.
  ```

- For a specific HP-UX hardware path, if the EFI path mapping is stale or not up to date, then booting from the disk will fail with an error message similar to the following:

  ```
  Start of HP-UX HA Alternate Boot: 1/0/0/2/0.6.0 failed:
  Not Found
  ```

`vparefiutil` without any options can be used to display the current hardware to EFI path mappings.

For more information on `vparefiutil` and all the possible options, see the manpage *vparefiutil* (1M).

NOTE Following are some scenarios where you may need to perform additional actions if the EFI path to hardware path mappings are not up to date in the vPars database:

- Creating an alternate vPars database while in `vPars` mode.

  Problem:

  If an alternate vPars database is created while in `vPars` mode and the Monitor is later booted using that alternate database, then it may not be possible to boot some of the virtual partitions of the alternate database if the EFI paths corresponding to those hardware paths are not present in the alternate database.

  Solutions:

  - The virtual partitions that could not boot can be re-installed using `vparboot -I`:

    `vparboot -p` *`partition_name`* `-I`

  - The virtual partitions that could not boot can be booted using the Monitor command `vparload`:

    ```
    vparload -p partition_name -E disk_index
    ```

- Creating a virtual partition in `vPars` mode.

  Problem:

  If a virtual partition is created while in `vPars` mode, then it may not be possible to boot that partition if the EFI path corresponding to the boot disk hardware path is not present in the vPars database.

  Solutions:

  - The virtual partition can be re-installed using `vparboot -I`:

    ```
    vparboot -p partition_name -I
    ```

  - The virtual partition can be booted using the Monitor command `vparload`:

    ```
    vparload -p partition_name -E disk_index
    ```

- An OS is installed *not* using `vparboot -I` and the database is created as a last step.

  Problem:

  If an OS is installed on one disk (for example, vpar1), the database (`vpdb`) is created on vpar1, an OS is installed on vpar2 in `nPars` mode, `vparenv` is executed on vpar2 to change the mode to `vPars`, and the Monitor is booted from the boot disk of vpar1, then it may not be possible to boot vpar2.

  Solutions:

  - Boot from vpar1's boot disk into `nPars` mode and execute the following set of commands to update the vPars database and change the mode:

    ```
    vpar1# vparefiutil -u [-D /stand/vpdb]
    vpar1# vparenv -m vPars
    ```

  - Create the database (`vpdb`) on the last installed virtual partition and boot the Monitor from it.

- MirrorDisk and EFI path.

  Problem:

  If the `idisk` command is executed on a disk during mirror disk creation, the EFI path of the disk may change. It may not be possible to then boot from the new mirrored disk using `vparboot -B`.

  Solutions:

  - After creating the mirror disk, set the mirror disk as alternate path using `setboot`.

    ```
    # setboot -a mirror_disk_hw_path
    ```

  - Execute the `vparefiutil` command on the new disk.

    ```
     # vparefiutil -u [-H mirror_disk_hw_path]
    ```

- Booting from a recently added boot disk.

  Problem:

If you add a boot disk at a known hardware path, it may not be possible to immediately boot from this new disk.

Solution:

- If the EFI signature of the disk is known, the `vparload -E` command can be used to boot from the disk.

CAUTION It is recommended to use the documented procedure of using `vparboot -I` to create the virtual partitions so that users do not have to use `vparload -E`. For information on the Monitor command `vparload`, see the Monitor manpage *vpmon* (5).

`vparload -E` works in a trial and error fashion, meaning you may have to serially attempt different disk indices. If there are multiple boot disks belonging to a virtual partition, booting with `vparload -E` is easier if the EFI signature of the desired boot disk is known.

Attempting to boot from a incorrect boot disk may result in an ungraceful shutdown or early panic with the message "`root already mounted`" if another OS instance is already booted from that boot disk.

Monitor: Booting the vPars Monitor

To boot the vPars Monitor, from ISL or EFI, specify `/stand/vpmon`:

— PA-RISC:

```
ISL> hpux /stand/vpmon
```

— Integrity:

```
Shell> fs0
fs0:\> hpux
HPUX> boot vpmon
```

Note: you must be in `vPars` mode to boot the Monitor. See “Modes: Switching between nPars and vPars Modes (Integrity only)” on page 95. Also, backspace is sometimes not parsed correctly; if the command fails, try again without backspacing.

With no arguments to `vpmon`, the Monitor will load and go into interactive mode with the following prompt:

```
MON>
```

The following options are available when booting the Monitor:

`-a`	boots all virtual partitions that have the autoboot attribute set. For more information, see *vparmodify* (1M).
`-D` *`database_filename`*	boots the virtual partitions using an alternate partition database file. For more information, see “Using an Alternate Partition Database File” on page 153. The default partition database file is `/stand/vpdb`.

For more information on the vPars boot sequence, see “Boot Sequence” on page 33.

Monitor: Accessing the Monitor Prompt

You can reach the Monitor prompt in the following ways:

- From the ISL or EFI prompt, you can boot the Monitor into interactive mode (see "Monitor: Booting the vPars Monitor" on page 103).
- After shutting down all virtual partitions, you will arrive at the Monitor prompt on the console (see "Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)" on page 135).
- A.03.xx and earlier: When the system monarch CPU is not owned by any virtual partition, you will also see the Monitor prompt `MON>` while toggling among the virtual consoles.

  A monarch CPU exists in both non-vPars and vPars servers. After a server is powered-on, the monarch CPU determines what other CPUs are configured in the server and then launches the other CPUs to create a multi-CPU server. Typically, the CPU with the lowest numbered hardware path address (belonging to the core cell for nPartitionable systems) is the monarch CPU. To see the lowest numbered hardware path, on a non-vPars server use `ioscan`, or on a vPars server use the Monitor command `scan`.

  A.04: When any CPU is available, you will see the MON> prompt.

Monitor: Using Monitor Commands

You can use the following Monitor commands at the Monitor prompt for booting and basic troubleshooting. However, most vPars operations should be performed using the vPars shell commands.

Note the following for the Monitor commands:

- Unless specifically stated, all operations occur only on the boot disk from which the Monitor was booted. Usually, this is the boot disk of the primary path entry in system-wide stable storage.

  Further, the Monitor can traverse only HFS file systems. Usually, the only HFS file system is `/stand`.

- Except for the `vparload` command, an alternate disk device cannot be specified using the Monitor commands.

- The following Monitor commands are disabled when one or more virtual partitions are up:

  `getauto`, `lifls`, and `readdb`.

- The following Monitor commands are disabled when the virtual partition that owns the disk from which the Monitor was booted, usually the primary path, is up:

  `ls` and `cat`.

NOTE You can see all the latest Monitor commands and options from the *vpmon* (1) manpage.

Not all Monitor commands are available on all platforms. The following common Monitor commands are not available on Integrity systems: `cat`, `cbuf`, `getauto`, `lifls`, and `ls`. Refer to the *vpmon* (1) manpage for a complete list.

Booting

- **`readdb`** ***`filename`***

  reads an alternate partition database *`filename`* for partition configuration information

  `filename` must be an absolute path and reside on a HFS file system.

  Example:

  If you have a backup copy of the partition database in the file `/stand/vpdb.backup`, you can read the database configuration information using:

  ```
  MON> readdb /stand/vpdb.backup
  ```

  Notes:

  This command can only be used when the Monitor `/stand/vpmon` is booted and the default partition database (`/stand/vpdb`) does not exist, the alternate partition database as specified in the `-p` option of `/stand/vpmon` does not exist, or the database file read is corrupt. For more information on the -p option, see “Monitor: Booting the vPars Monitor” on page 103.

  Integrity only: If you issue `readdb /stand/vpdb.backup`, the file that is actually read is at `/stand/boot.sys/stand/vpdb.backup`. The `vparcreate` command transparently creates the soft link from `/stand/boot.sys/stand/`*`file`* to `/stand/`*`file`*. Therefore, if you backup the database file using the Unix `cp` command, the `ln -s` command also should be executed to create the soft link. Otherwise, it will not be possible to boot from the backup database file.

- **`vparload -all`**
  `vparload -auto`
  `vparload -p` ***`partition_name`*** **[`-b`** ***`kernelpath`*****] [`-o`** ***`boot_options`*****] [`-B`** ***`hardware_path`*****]**

  boots the virtual partition *`partition_name`*; this command is similar to the vPars Unix shell command `vparboot.`

`-all`	boots all virtual partitions, regardless of the `autoboot` or `autosearch` attributes. For more information on the `autoboot` or autosearch attributes, see the *vparcreate* (1M) or *vparmodify* (1M) manpages.
`-auto`	boots all virtual partitions that have their autoboot attribute flag set to `AUTO`.
`-b` *`kernelpath`*	boots the virtual partition using the kernel *`kernelpath`* instead of the default kernel
`-o` *`boot_options`*	boots the virtual partition using the options *`boot_options`*, such as `-is` for single-user mode or `-lm` for LVM maintenance mode.
`-B` *`hardware_path`*	boots the virtual partition using the disk device at the *`hardware_path`*

Examples:

To boot the partition `winona2` into single-user mode:

```
MON> vparload -p winona2 -o "-is"
```

To boot the partition `winona2` using the kernel `/stand/vmunix.other`:

```
MON> vparload -p winona2 -b /stand/vmunix.prev
```

To boot the partition `winona2` using the disk device at `0/8/0/0.2.0`:

```
MON> vparload -p winona2 -B 0.8.0.0.2.0
```

Note:

`-b` *`kernelpath`* allows you to change the target kernel for only the next boot of *`partition_name`*. If you wish to make a permanent change to the partition database, use the `vparmodify` command.

For example, to change the partition database information so that `winona2` always boots using `/stand/vmunix.other`:

```
# vparmodify -p winona2 -b /stand/vmunix.other
```

See the *vparmodify* (1M) manpage for more information on modifying the partition database.

(vPars A.04.01) For 11i v2 (11.23) systems, alternate kernels are in the directory `/stand/alternate_config/`.

Also, when a virtual partition is booted, there may be a pause in the console output. For more information, see “Boot||Shut: Booting a Virtual Partition” on page 132.

Finally, when there is a pending reboot for reconfiguration for the involved nPartition, the target virtual partitions will not be booted until all the virtual partitions within the nPartition have been shut down and the vPars Monitor rebooted. For more information see “Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)” on page 135.

- **bootpath**

  displays the device from which the vPars Monitor (/stand/vpmon) was booted

  Example:

  ```
  MON>bootpath
  disk(0.0.2.0.6.0)
  ```

- **reboot [*mode*]**

  reboots the entire hard partition. Other hard partitions are not affected.

  `mode` sets the mode for the next reboot and has the value of either `nPars` or `vPars`. This is applicable on only Integrity systems.

NOTE You should shut down each virtual partition (using the Unix `shutdown` command) prior to executing the Monitor `reboot` command. A confirmation prompt is provided, but if you accept confirmation of the reboot while any virtual partitions are running, the `reboot` brings the running partitions down ungracefully. For more information, see “Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)” on page 135.

Displaying Information

- **cat *filename* [openonly]**

  displays the contents of *`filename`*. When `openonly` is specified, this command only prints "`open succeeded`" if the Monitor was able to open the *`filename`*. This command is similar to the Unix `cat` command.

  `filename` must be a text file on an HFS file system.

  /`stand` is the default directory

  Example:

  To display the file `/stand/notes.txt`

  ```
  MON> cat notes.txt
  10/13/2001: built new kernel today. if problems arise, revert to saved kernel
  vmunix.original
  ```

- **cbuf *partition_name***

  displays the contents of the console buffer of *`partition_name`*

- **help**

  `help` or `?` lists all Monitor commands

- **lifls**

  lists the files in the LIF area

- **getauto**

  displays the contents of the `AUTO` file in the LIF area

Example:

```
MON> getauto
hpux /stand/vpmon
```

- **`log`**

  displays the contents, including warning and error messages, of the Monitor log. The Monitor log holds up to 16KB of information in a circular log buffer. The information is displayed in chronological order.

- **`ls [-alniFH] [`*`directory`*`]`**

  lists the contents of *`directory`*. This command is similar to the Unix `ls` command.

  `directory` must be on a HFS file system. `/stand` is the default directory

  The `ls` command-line options are the same as the Unix shell `ls` options. For detailed explanations, see the *ls* (1M) manpage. In brief:

-a	all entries
-l	long listing
-n	numerical UIDs and GIDs
-i	inode
-F	appends a character after the entry, depending on the file type, such as a `/` (slash) for a directory

  For example, to view the listing of files in `winona2`'s `/stand` directory:

```
MON> ls /stand
lost+found      ioconfig        bootconf        system
system.d        vmunix          dlkm.vmunix.prevbuild
kernrel         rootconf        vpdb            vpmon.dmp
vmunix.backup   system.prev     vmunix.prev     dlkm
vpdb.backup     vpmon
```

- **`scan`**

  lists all hardware discovered by the Monitor and indicates which virtual partition owns each device.

- **`toddriftreset`**

  resets the drifts of the real-time clock. Use this command if you reset the real-time clock of the hard partition at the BCH prompt. For brief information, see "Real-time clock (RTC)" on page 25.

- **`time`**

  displays systems real time clock and OS time of all the virtual partitions in GMT (Greenwich Mean Time). The OS time displayed will consider the RTC and clock drift for the virtual partition. However, if the partition is up, there may be difference in the OS time displayed.

- **`settime [`*`MM DD YYYY hh mm ss`*`]`**

  sets the system's real time clock. Acceptable date range is 1-1-1970 00:00:00 to 12-31-2034 23:59:59.

- **`vparinfo`** ***`[partition_name]`***

  This command is for HP internal use only.

  when no *`partition_name`* is given, `vparinfo` displays all unassigned resources and the names of all existing virtual partitions; when *`partition_name`* is given, `vparinfo` displays the resources assigned to *`partition_name`*.

Monitor: Using the Monitor Commands from ISL or EFI

You can specify any of the Monitor commands either at the Monitor prompt (MON>) or at the ISL prompt (ISL>). If you are at ISL or EFI, use the desired command as the argument for the Monitor `/stand/vpmon`.

For example, to run the command `vparload -p winona1` from the Monitor prompt, use

```
MON> vparload -p winona1
```

To run the same command from ISL or EFI, use

```
ISL> hpux /stand/vpmon vparload -p winona1
```

```
Shell> fs0:efi\hpux\hpux boot vpmon vparload -p winona1
```

where the command `(vparload -p winona)` is the argument for the Monitor `(/stand/vpmon)`.

Commands: vPars Manpages

The purpose of this document is to describe vPars concepts and how to perform common vPars tasks. For detailed information on the vPars commands, including description, syntax, all the command line options, and the required state of a virtual partition for each command, see the vPars manpages.

NOTE The *vparresources* (5) manpage contains critical information about the vPars commands, including option precedence and the required state of a virtual partition prior to command execution. The *vparcreate* (1M) and *vparmodify* (1M) manpages contain autoboot information.

Note the following on vPars manpages:

— If your OS is setup to use manpage keywords and vPars is installed, you can run the following to see the current list of vPars manpages.

```
# man -k vpar
```

For more information on manpage keywords and using `catman -w`, see the manpage *catman* (1M) and the manual "Managing Systems and Workgroups".

— As of this printing, the vPars manpages are:

vecheck (1), *vparboot* (1M), *vparconfig* (1M), *vparcreate* (1M), *vparefiutil* (1M), *vparenv* (1M), *vparmodify* (1M), *vparremove* (1M), *vparreset* (1M), *vparstatus* (1M), *vparutil* (1M), *vparresources* (5), *vpartition* (5), *vpmon* (1)

Commands: vPars Commands Logging

Beginning with vPars A.03.02, vPars will log the vPars commands executed from the HP-UX shell to the local syslog file (the syslog file of the virtual partition from which the vPars command was executed).

Log File Location and Log Format

The default syslog file on HP-UX systems is `/var/adm/syslog/syslog.log`

The format of the log entries is

```
date hostname vPars_command_name[pid]: user username: vPars_command_line_text
date hostname vPars_command_name[pid]: exit status exit_status
```

where

vPars_command_name is the name of the vPars command which is sending messages to syslog.

username is name returned by `getlogin()`. If no username is given by `getlogin()`, the effective username or id will be used.

vPars_command_line_text is the vPars command line text as typed by the user.

pid is the pid (Process ID) of the command invocation. The PIDs shown will be the same for both the command invocation syslog entry and the exit status syslog entry. This allows matching the exit status with its corresponding command invocation.

Below are examples of vPars syslog entries.

```
Oct 29 19:44:30 winona2 vparutil[2947]: user root: vparutil -s 1/0/0/3/1.7.0 -i 7
Oct 29 19:44:30 winona2 vparutil[2947]: exit status 4
Oct 29 19:47:47 winona2 vparmodify[2962]: user root: /sbin/vparmodify -p winona3 -a cpu::1
Oct 29 19:47:47 winona2 vparmodify[2962]: exit status 1
```

Cases Where No Logging Occurs

Below are the cases where logging does *not* occur:

- a non-root user attempting a vPars command
- syntax, usage, or vPars commands version errors
- the user replies "no" to vPars commands that request a confirmation before execution
- vPars commands which do not change the vPars database and/or do not affect the state of other virtual partitions. These commands include `vparstatus`, `vparextract`, `vecheck`, `vpardump`, and `vparreloc`. Additionally, read-only requests, such as `vparutil -g` (get), will not be logged.

Cases Where Logging Occurs

Below are the cases where logging does occur:

- vPars command which change the vPars database and/or affect the state of other virtual partitions. These commands include `vparboot`, `vparcreate`, `vparremove`, `vparmodify`, `vparreset`, and `vparutil`.

Constraints and Restrictions to Logging

Note the following:

- Commands will be logged whether executed on the vPars database in memory, an alternate database, or in standalone mode.
- The command line text will be logged on only the partition from which the command was executed. The logging of the command will not be *duplicated* to the target syslog file (the syslog file of the target virtual partition.
  For example, if the `vparmodify` command is executed from `winona1` with the target partition being `winona2` (`winona1# vparmodify -p winona2 ...`), the syslog entries will only appear in the log file of `winona1`. Nothing will appear in the log file of `winona2`.
- Interaction with the user, including error messages or requests for confirmation, are not logged.
- When confirmation to execute a vPars command is requested, but the user replies “no”, the vPars command is not logged.

Syslog and Priority and Facility Codes

You can continue to use priority and facility codes of syslogd to configure vPars logging. vPars uses LOG_INFO as the priority level and LOG_USER as the facility. For more information on using the priority and facility codes of syslogd, see the manpages *syslogd* (1M) and *syslog* (3C).

NOTE nPartition Logs

On an nPartition server running vPars, all virtual partitions within an nPartition share the same console device: the **nPartition’s console**. Thus, an nPartition’s console log contains console IO for multiple virtual partitions. Further, since the vPars Monitor interface is displayed and accessed through the nPartition’s console, vPars Monitor output is also recorded in the nPartition’s console log. There is only one Monitor per nPartition.

The **server chassis logs** record nPartition and server complex hardware events. The chassis logs do not record vPars-related configuration or vPars boot events; however, the chassis logs do record HP-UX "heartbeat" events. The server chassis logs are viewable from the GSPs Show Chassis Log menu. For more information, see the Help within the GSPs online help.

The **vPars Monitor event logs** record only vPars events; it does not contain any nPartition chassis events. For more information, see *vparstatus* (1M).

Also, for a given nPartition, the Virtual Front Panel (VFP) of the nPartition’s console displays an OS heartbeat whenever at least one virtual partition within the nPartition is up.

Commands: Displaying Monitor and Resource Information (vparstatus)

The Monitor and the partition database maintains information about all the virtual partitions, including the current state of the virtual partitions and their resources. Using the shell command `vparstatus`, you can display this information. This section describes the possible virtual partition states and the common usages of the vparstatus command.

Virtual Partition States

Virtual partitions can be in the following states:

Table 5-2 Virtual Partition States

State	Description
`load`	The Monitor is loading the kernel image of the virtual partition. This is the first step of booting a virtual partition. If successful, the state moves to `boot`.
`boot`	The Monitor has successfully loaded the kernel image and is continuing with the boot process. If the launch is successful, the state moves to `up`.
`up`	The virtual partition is up and running.
`shut`	The virtual partition is shutting down gracefully. Once the partition is shutdown, the state moves to `down`.
`down`	The virtual partition is down.
`crash`	The virtual partition is crashing because of a panic (HPMC, TOC, etc.). Once the partition has completed crashing, the state moves to `down`.
`hung`	The virtual partition is not responding.
`N/A`	The virtual partition is in a database file that is not active, so it has no state. The database file can be inactive because either the system is in standalone mode (the vPars Monitor is not running) or the database file acted upon is an alternate database file that is not in Monitor memory.

`vparstatus` output examples

The next few pages show examples of using the `vparstatus` command. For detailed usage, syntax, and information, see the manpage *vparstatus* (1M).

NOTE Actual `vparstatus` output differs from version to version of vPars. Depending on your version and system, the output shown below may differ.

The kernel path shown in the `vparstatus` output is the kernel path set in the vPars database. This may differ from the actual kernel in use.

vparstatus: summary information

To see summary information on all the virtual partitions, use **vparstatus with no options**:

- vPars A.03.xx on a rp7400:

```
winona1# vparstatus
[Virtual Partition]
                                                                        Boot
Virtual Partition Name         State Attributes Kernel Path               Opts
============================== ===== ========== ========================= =====
winona1                        Up    Dyn,Auto   /stand/vmunix
winona2                        Up    Dyn,Auto   /stand/vmunix
winona3                        Up    Dyn,Auto   /stand/vmunix

[Virtual Partition Resource Summary]
                                            CPU    Num        Memory (MB)
                                   CPU     Bound/   IO   # Ranges/
Virtual Partition Name           Min/Max  Unbound  devs  Total MB    Total MB
==============================   ================  ====  ====================
winona1                            2/  8    2   0     2    0/  0         1024
winona2                            2/  8    2   1     2    0/  0        1280
winona3                            1/  8    1   0     2    0/  0        1280
```

- vPars A.04.xx on an Integrity Superdome:

```
thurman24# vparstatus
[Virtual Partition]
                                                                          Boot
Virtual Partition Name          State Attributes   Kernel Path             Opts
=============================== ===== ============ ======================= =====
thurman24                       Up    Dyn,Auto,Nsr /stand/vmunix
thurman25                       Up    Dyn,Auto,Nsr /stand/vmunix
thurman26                       Up    Dyn,Auto,Nsr /stand/vmunix
thurman27                       Up    Dyn,Auto,Nsr /stand/vmunix

[Virtual Partition Resource Summary]
                                CPU      Num   Num     Memory Granularity
Virtual Partition Name          Min/Max  CPUs  IO       ILM         CLM
==============================  =======  ====  ====  ==========  ==========
thurman24                         1/ 28     1     3         128         128
thurman25                         1/ 28     6     2         128         128
thurman26                         1/ 28     3     3         128         128
thurman27                         1/ 28     3     4         128         128

                                                  Memory (MB)
                                          ILM                    CLM
                                # User                   # User
Virtual Partition Name          Ranges/MB     Total MB  Ranges/MB     Total MB
==============================  ======================  ======================
thurman24                         0/    0         1024    0/    0            0
thurman25                         1/1024          1024    0/    0            0
thurman26                         0/    0         1024    1/ 256           256
thurman27                         1/ 512          1024    1/ 768          1024
```

vparstatus: verbose information

To see **verbose (-v)** information:

- vPars A.03.xx on a rp7400:

```
winona1# vparstatus -p winona2 -v

[Virtual Partition Details]
Name:         winona2
State:        Up
Attributes:   Dynamic,Autoboot
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/8
Bound by User [Path]:  41
                       45
Bound by Monitor [Path]:
Unbound [Path]:  97

[IO Details]
    0.8.0.0.5.0  BOOT
    0.8
    1.10

[Memory Details]
Specified [Base  /Range]:
           (bytes) (MB)
Total Memory (MB):  1280
```

- vPars A.04.xx on an Integrity Superdome:

```
thurman24# vparstatus -p thurman25 -v

[Virtual Partition Details]
Name:         thurman25
State:        Up
Attributes:   Dynamic,Autoboot,Nosearch
Kernel Path:  /stand/vmunix
Boot Opts:

[CPU Details]
Min/Max:  1/28
User assigned [Path]:
Boot processor [Path]:  13.120
Monitor assigned [Path]:  1.121
                          1.122
                          3.122
                          3.123
                          4.121

Non-cell-specific:
  User assigned [Count]:     1
  Monitor assigned [Count]:  3
Cell-specific [Count]:  Cell ID/Count
                              1     2

[IO Details]
   1.0.12
   1.0.12.1.0.4.0.8.0.255.0.2.0 BOOT

[Memory Details]
ILM, user-assigned [Base  /Range]:  0x100000000/1024
                   (bytes) (MB)
```

```
ILM, Monitor-assigned [Base  /Range]:
                      (bytes) (MB)
ILM Total (MB):  1024

ILM Granularity (MB):  128

CLM, user-assigned [CellID Base  /Range]:
                          (bytes) (MB)
CLM, Monitor-assigned [CellID Base  /Range]:
                             (bytes) (MB)
CLM (CellID MB):

CLM Granularity (MB):  128
```

vparstatus: available resources

To see the **available resources (-A)** (resources not assigned to any virtual partition):

- A.03.xx on a rp7400 (non-nPartitionable server)

```
winona1# vparstatus -A
[Unbound CPUs (path)]:  101
                        109
[Available CPUs]:  2

[Available IO devices (path)]:  1.2

[Unbound memory (Base  /Range)]:  0x40000000/256
                 (bytes) (MB)
[Available memory (MB)]:  256
```

- A.04.xx on a rp7420 (nPartitionable server)

```
keira1# vparstatus -A
[CPUs (path)]:  0.11
                0.12
                0.13
                1.11
                1.13
                1.14
[CLP (CellID Count)]:  0 3
                       1 3
[Available CPUs]:  5

[Available I/O devices (path)]:  0.0.0
                                 0.0.1
                                 0.0.4
                                 0.0.6
                                 0.0.10
                                 0.0.12
                                 0.0.14
                                 1.0.1
                                 1.0.2
                                 1.0.4
                                 1.0.6
                                 1.0.8
                                 1.0.10
                                 1.0.14

[Available ILM (Base  /Range)]:  0x0/256
               (bytes) (MB)      0x18000000/2560
                                 0xf0000000/2304
[Available ILM (MB)]:  4096

[Available CLM (CellID Base  /Range)]:  0 0x70080000000/2048
                      (bytes) (MB)
[Available CLM (CellID MB)]:  0 2048
                              1 0
```

vparstatus: pending nPartition RFR

On an nPartitionable system, if the nPartition has a pending RFR (Reboot-for-Reconfig), the `vparstatus` output will show the following message:

```
Note: A profile change is pending.  The hard partition must be rebooted to complete it.
```

- Example:

```
keira1# vparstatus
[Virtual Partition]
                                                                           Boot
Virtual Partition Name          State Attributes Kernel Path               Opts
============================== ===== ========== ========================= =====
keira1                         Up    Dyn,Auto   /stand/vmunix
keira2                         Down  Dyn,Manl   /stand/vmunix             boot

[Virtual Partition Resource Summary]
                                           CPU    Num        Memory (MB)
                                  CPU     Bound/   IO   # Ranges/
Virtual Partition Name          Min/Max  Unbound  devs  Total MB    Total MB
==============================  ================  ====  ====================
keira1                            2/  8     2    0     7    0/  0        2048
keira2                            2/ 12     2    2     3    0/  0        2048

Note:  A profile change is pending.  The hard partition must be rebooted to complete it.
```

`vparstatus`: Monitor and database information

Beginning with vPars A.03.02, the **-m** option displays the console path, the hardware path from which the Monitor was booted, the filesystem path of the Monitor, and the vPars database file that is being used by the Monitor:

- vPars A.03.02:

```
# vparstatus -m
Console path: 0.0.2.0
Monitor Boot disk path: 0.0.1.0
Monitor Boot filename: /stand/vpmon
Database filename: /stand/vpdb.mine
```

- vPars A.04.01:

```
# vparstatus -m
Console path:  No path as console is virtual
Monitor boot disk path:  13.0.11.1.0.8.0
Monitor boot filename:  /stand/vpmon
Database filename:  /stand/vpdb
Memory ranges used:  0x0/232611840 Monitor
                     0xddd6000/688128 firmware
                     0xde7e000/1384448 Monitor
                     0xdfd0000/33751040 firmware
                     0x10000000/134213632 Monitor
                     0x7fffe000/8192 firmware
                     0x8a0ff000000/16777216 firmware
```

vparstatus: migrating CPUs

Migrating CPUs may not occur instantaneously. If a virtual partition has a pending (in other words, still in progress) addition or deletion of one or more CPUs, the letter `p` will be displayed next to the number of CPUs in the summary output and the words `(migration pending)` will be displayed in the detailed output:

-
  ```
  winona1# vparstatus
  .
  .
  .
  [Virtual Partition Resource Summary]
                                             CPU    Num        Memory (MB)
                                    CPU     Bound/   IO   # Ranges/
  Virtual Partition Name          Min/Max  Unbound  devs  Total MB    Total MB
  ==============================  ================  ====  ====================
  winona1                           2/  8    2   0      2    0/  0         1024
  winona2                           2/  8    2   1p     2    0/  0        1280
  winona3                           1/  8    1   0      2    0/  0        1280


  winona1# vparstatus -p winona2 -v
  ...
  [CPU Details]
  Min/Max:  1/8
  Bound by User [Path]:  41
                         45
  Bound by Monitor [Path]:
  Unbound [Path]:  97 (migration pending)

  [IO Details]
      0.8.0.0.5.0  BOOT
      0.8
      1.10

  [Memory Details]
  Specified [Base  /Range]:
             (bytes) (MB)
  Total Memory (MB):  1280
  ```

vparstatus: dual-core CPUs

You can see the sibling and virtual partition assignment using `vparstatus` **`-d`**. If you do not have a dual-core system, the output will show dashes (-) for the sibling and assignment information:

```
# vparstatus -d
CPU                       Cell Config                   Sibling Information
path  CPU HPA             ID   Status Assigned to       Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc078000    0 E      vpuma02            -     -
0.11  0xfffffffffc07a000    0 E      vpuma01            -     -
0.12  0xfffffffffc07c000    0 E      vpuma04            -     -
0.13  0xfffffffffc07e000    0 E      -                  -     -
...
```

When you do have dual-core system, the `vparstatus -d` output will look similar to the following:

```
# vparstatus -d
CPU                       Cell Config                   Sibling Information
path  CPU HPA             ID   Status Assigned to       Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc070000    0 E      vpkeira1           0.11  vpkeira3
0.11  0xfffffffffc071000    0 E      vpkeira3           0.10  vpkeira1
0.12  0xfffffffffc074000    0 E      -                  0.13  vpkeira4
0.13  0xfffffffffc075000    0 E      vpkeira4           0.12  -
0.14  0xfffffffffc078000    0 E      -                  0.15  -
0.15  0xfffffffffc079000    0 E      -                  0.14  -
...
```

`vparstatus`: CPU information on vPars A.04

While a virtual partition is in the down state, no specific CPU is assigned to the virtual partition as the boot processor but one is allocated by the Monitor if needed (there are no CPUs assigned to the virtual partition). The boot processor is determined when the virtual partition is booted.

Note that the output does not determine which commands were issued; different commands can be used to arrive at the same vparstatus output.

Table 5-3 possible commands to arrive at vparstatus output

vparstatus output (final)	set of possible commands in sequence to create vparstatus output
keira1 # vparstatus -p keira1 -v [Virtual Partition Details] Name: keira1 State: Up Attributes: Dynamic,Autoboot,Autosearch Kernel Path: /stand/vmunix Boot Opts: [CPU Details] Min/Max: 1/12 User assigned [Path]: 1.10 Boot processor [Path]: 1.12 Monitor assigned [Path]: 0.10 0.11 1.11 Non-cell-specific: User assigned [Count]: 2 Monitor assigned [Count]: 2 Cell-specific [Count]: Cell ID/Count 1 1 ...	# vparcreate -p keira1 -a cpu:::1:12 -a cpu::4 (min==1, max==12, total==4 4 non-CLPs are reserved by the Monitor) # vparmodify -p keira1 -a cpu:1/10 -a cpu:1/12 (since the vpar is down, total is not modified. therefore, 2 of the 4 CPUs assigned by the Monitor become user-assigned by hardware_path (1.10 and 1.12). total==4 (2 assigned by hardware path and 2 assigned by Monitor)) # vparmodify -p keira1 -a cell:1:cpu::1 (since the specification is by CLP, total is modified. the Monitor chooses which CPU from cell 1. totat==5 (2 assigned by hardware path and 2 assigned by Monitor and 1 assigned by CLP)) # vparboot -p keira1 assume IO and memory have been assigned so that keira1 can boot. at boot time: min==1, max==12 user-assigned CPU is 1.10 boot processor is chosen by Monitor to be the other user-assigned CPU at 1.12 2 CPUs assigned by Monitor are 0.10 and 0.11 1 CPU assigned by the Monitor by CLP is 1.11

Table 5-4 possible commands to arrive at vparstatus output

vparstatus output (final)	another set of possible commands in sequence to create vparstatus output
keira1 # vparstatus -p keira1 -v [Virtual Partition Details] Name: keira1 State: Up Attributes: Dynamic,Autoboot,Autosearch Kernel Path: /stand/vmunix Boot Opts: [CPU Details] Min/Max: 1/12 User assigned [Path]: 1.10 Boot processor [Path]: 1.12 Monitor assigned [Path]: 0.10 0.11 1.11 Non-cell-specific: User assigned [Count]: 2 Monitor assigned [Count]: 2 Cell-specific [Count]: Cell ID/Count 1 1 ...	# vparcreate -p keira1 -a cpu:1.12 (total==1) # vparboot -p keira1 (assume IO and memory have been assigned so that keira1 can boot. at this point the cpu assigned by hw_path (1.12) is the only CPU, so it becomes the boot processor. Note that 1.12 is only listed once; it is *not* listed under both "Boot Processor" and "User Assigned") # vparmodify -p keira1 -a cpu:1.10 (1.10 is added and listed as "user assigned") # vparmodify -p keira1 -a cpu::2 (0.10 and 0.11 are chosen by the Monitor and added) # vparmodify -p keira1 -a cell:1:cpu::1 (1.11 is added and listed under "Cell-specific")

Managing: Creating a Virtual Partition

You can create a virtual partition using the `vparcreate` command.

NOTE When you create a virtual partition, the vPars Monitor assumes you will boot and use the partition. Therefore, when a virtual partition is created, even if it is down and not being used, the resources assigned to it cannot be used by any other partition.

Also, when using vPars, the physical hardware console port must be owned by a partition. To avoid terminal type mismatches, this should be the first virtual partition created. For an example, see "Ensuring the Hardware Console Port Is Owned by the First Virtual Partition (PA-RISC)" on page 55.

For memory considerations, please see "Memory: Allocation Concepts and Notes" on page 178.

CAUTION *LBAs must be explicitly specified when using vPars A.03.01 or earlier*. The examples in this chapter use a non-nPartitionable system. If using an nPartitionable system, please read "Planning, Installing, and Using vPars with an nPartitionable Server" on page 48.

Example (A.03.xx)

To create a virtual partition named `winona2` with the following resources:

- Three total CPUs (two bound CPUs at hardware paths 41 and 45 and one unbound CPU) with a maximum of four (bound plus unbound) CPUs
- 1280 MB of memory
- all hardware where the LBA is at `0/8` or `1/10`
- a boot disk at `0/8/0/0.5.0`

use the corresponding `vparcreate` command line options:

Resource or Attribute	vparcreate Option
virtual partition name is winona2	`-p winona2`
three total CPUs	`-a cpu::3`
of which two are bound CPUs and a maximum of four CPUs	`-a cpu:::2:4`
at hardware paths 41 and 45	`-a cpu:41 -a cpu:45`
1280 MB of memory	`-a mem::1280`
all hardware where the LBA is at 0/8	`-a io:0.8`
all hardware where the LBA is at 1/10	`-a io:1.10`
hardware at 0/8/0/0.5.0 as the boot disk	`-a io:0.8.0.0.5.0:boot`

The resulting `vparcreate` command line is:

```
winona1# vparcreate -p winona2 -a cpu::3 -a cpu:::2:4 -a cpu:41 -a cpu:45 -a mem::1280 -a
io:0.8 -a io:1.10 -a io:0.8.0.0.5.0:boot
```

TIP For the `vparcreate` options, you can create a text file that includes all the options and then `cat` the text file within the `vparcreate` command line. This avoids having to remember all the options when you are typing the `vparcreate` command line.

For example, for the vPars A.03.xx `vparcreate` command of `winona2`, you can create a text file called `/stand/winona2.opts`:

```
winona1# vi /stand/winona2.opts
```

The text file would contain the following:

```
-p winona2 \
-a cpu::3 \
-a cpu:::2:4 \
-a cpu:41 \
-a cpu:45 \
-a mem::1280 \
-a io:0.8 \
-a io:1.10 \
-a io:0.8.0.0.5.0:boot
```

When you are ready to execute the `vparcreate` command, the command appears as:

```
winona1# vparcreate `cat /stand/winona2.opts`
```

You can verify the creation using the `vparstatus` command:

```
winona1# vparstatus -p winona2 -v
```

Managing: Removing a Virtual Partition

To remove a virtual partition, use `vparremove`. `vparremove` purges the virtual partition from the vPars database. Any resources dedicated to the virtual partition are now free to allocate to a different virtual partition (for A.03, see Appendix B for exceptions).

You need to shutdown the virtual partition before attempting removal. If the target virtual partition is running, `vparremove` will fail.

Example

To remove a virtual partition named `winona2`:

1. If the virtual partition is running, shutdown the virtual partition:

```
winona2# vparstatus
winona2# shutdown -h
```

2. From the running virtual partition `winona1`, verify the target virtual partition `winona2` has entered the `down` state (for more information on virtual partition states, see “Commands: Displaying Monitor and Resource Information (vparstatus)” on page 114):

```
winona1# vparstatus | grep winona2
winona2         Down  Dyn,Auto   /stand/vmunix
winona2             2/  8    2   1     2    0/  0        1280
```

3. After the virtual partition is in the `down` state, remove the virtual partition `winona2`:

```
winona1# vparremove -p winona2
```

TIP When a virtual partition is removed, data residing on the disk(s) of the target partition is not removed. If you have removed a partition by accident, you may be able to recover the partition by immediately re-creating the same virtual partition with the same assigned resources.

NOTE If the `vparremove` fails but `vparstatus` shows the target virtual partition as down, please try the `vparremove` again after waiting a few seconds. There is a small window of time after a virtual partition is downed by the `shutdown` or `vparreset` command before you can perform the `vparremove` command successfully.

Managing: Modifying Attributes of a Virtual Partition

You can change a virtual partition's name and its resource attributes via the `vparmodify` command. When using `vparmodify` to change attributes, the partition can be running, and the changes take effect immediately. See the manpage *vparmodify* (1M) for more information on the attributes.

For information on modifying resources, see "CPU, Memory, and IO Resources (A.04.xx)" on page 157.

Examples

- To rename the virtual partition `uma1` to `winona1`:

  ```
  # vparmodify -p uma1 -P winona1
  ```

- To set the `autoboot` attribute to `manual` for partition `winona1` (`manual` turns autoboot off. By default, the attribute is `auto`, which turns autoboot on):

  ```
  # vparmodify -p winona1 -B manual
  ```

- To set the `autoboot` attribute to `auto` and the `autosearch` attribute to search for partition `winona1`:

  ```
  # vparmodify -p winona1 -B auto -B search
  ```

- To set the `static` attribute for partition `winona1` (`static` disables modification of a partition's resources. By default, the attribute is `dynamic`):

  ```
  # vparmodify -p winona1 -S static
  ```

- To set the default kernel path to `/stand/vmunix.new` for the virtual partition `winona1`:

  ```
  # vparmodify -p winona1 -b /stand/vmunix.new
  ```

  (vPars A.04.01) For 11i v2 (11.23) systems, alternate kernels are in the directory `/stand/alternate_config/`.

Managing: Performing nPartition Operations

You can perform nPartition operations in a vPars environment, keeping in mind the following:

- If you make an nPartition change where a Reboot for Reconfiguration is required, all the virtual partitions within the nPartition need to be shutdown and the Monitor rebooted in order for the reconfiguration to take effect.
- Once the BIB (Boot-Is-Blocked) state is set on the nPartition, virtual partitions will not be able to boot up until all the virtual partitions have been shutdown and the Monitor rebooted.

For concepts on virtual partitions within nPartitions, see “nPartitions” on page 21.

Examples

For the following examples, the virtual partitions `keira1` and `keira2` exist within the nPartition `0`. Only relevant output is shown.

Reconfiguring an nPartition

1. Perform the changes as you would in a non-vPars environment. For example, if we want to add cell `6` to partition `0`:

```
keira1# parmodify -p0 -a 6:base:y:ri

In order to activate any cell that has been newly added,reboot the partition with the -R option.
Command succeeded.
```

2. Perform a Reboot-for-Reconfig (RFR) from a virtual partition. For example,

```
keira1# vparstatus
keira1# shutdown -R
.
.
.

Transition to run-level0 is complete.
Executing "/sbin/reboot-R   ".

Note: If this is a partitionable system, the requested reconfiguration will not take place until
all the virtual partitions on this hard partition are shut down and the virtual partition Monitor
is rebooted.

Shutdown at 16:09 (in 0 minutes)
```

At this point, the nPartition is in the Boot-Is-Blocked (BIB) state. The virtual partition `keira1` remains down until all the virtual partitions have been shutdown and the Monitor rebooted.

Note also that once the nPartition is in the BIB state, vparstatus shows the following message:

```
Note: A profile change is pending.  The hard partition must be rebooted to complete it.
```

3. Shutdown the other virtual partitions. For example:

```
keira2# vparstatus
keira2# shutdown -R
.
.
.
```

```
Transition to run-level0 is complete.
Executing "/sbin/reboot-R   ".

Note: If this is a partitionable system, the requested reconfiguration will not take place until
all the virtual partitions on this hard partition are shut down and the virtual partition Monitor
is rebooted.

Shutdown at 16:19 (in 0 minutes)
```

At this point, all virtual partitions have been shut down. The Monitor will reboot automatically. On the console, we would see the following message:

```
All partitions have halted.  System will now reboot for reconfiguration.
```

and the beginning of the boot process for the nPartition:

```
Firmware Version  21.3

Duplex Console IO Dependent Code (IODC) revision 2
   --------------------------------------------------------------------------
     (c) Copyright 1995-2002, Hewlett-Packard Company, All rights reserved
   --------------------------------------------------------------------------


          Cab/      Cell      ------- Processor --------    Cache Size
    Cell  Slot      State       #    Speed       State       Inst    Data
    ----  ----  ------------  ---  --------  -----------  ------  ------
      0   0/0   Idle           0A  1000 MHz  Idle         32 MB   32 MB
                               0B  1000 MHz  Idle         32 MB   32 MB
                               1A  1000 MHz  Idle         32 MB   32 MB
                               1B  1000 MHz  Idle         32 MB   32 MB
                               2A  1000 MHz  Idle         32 MB   32 MB
                               2B  1000 MHz  Idle         32 MB   32 MB
      1   0/1   Active         0A  1000 MHz  Active       32 MB   32 MB
                               0B  1000 MHz  Idle         32 MB   32 MB
                               1A  1000 MHz  Idle         32 MB   32 MB
                               1B  1000 MHz  Idle         32 MB   32 MB
                               2A  1000 MHz  Idle         32 MB   32 MB
                               2B  1000 MHz  Idle         32 MB   32 MB


     Primary Boot Path:  1/0/0/3/0.6
          Boot Actions:  Go to BCH.

HA Alternate Boot Path:  1/0/0/3/0.6
          Boot Actions:  Go to BCH.

   Alternate Boot Path:  1/0/12/1/0.8
          Boot Actions:  Go to BCH.

          Console Path:  1/0/0/0/1.0
```

Putting an nPartition into an Inactive State & Other GSP nPartition Operations

1. If possible, gracefully shutdown all the virtual partitions within the target nPartition. For example:

```
keira1# vparstatus
keira1# shutdown -h

keira2# vparstatus
keira2# shutdown -h
```

2. On the console, you will arrive at the MON> prompt. From the Monitor prompt, press `Ctrl-B` to enter into the GSP:

   ```
   MON> ^B
   GSP MAIN MENU:
    CO: Consoles
   VFP: Virtual Front Panel
    CM: Command Menu
    CL: Console Logs
    SL: Show chassis Logs
    HE: Help
     X: Exit Connection
   ```

3. At the GSP prompt, enter into the Command Menu

   ```
   GSP> cm

   Enter HE to get a list of available commands
   GSP:CM>
   ```

4. From the GSP Command Menu, perform the desired hard partition commands.

   For example, to make the hard partition and its cells inactive, use the RR (Reset for Reconfiguration) command:

   ```
   GSP:CM> rr

   This command resets for reconfiguration the selected partition.

   WARNING: Execution of this command irrecoverably halts all system processing andIO activity and
   restarts the selected partition in away that it canbe reconfigured.

   Do you want to reset for reconfiguration partition number 0? (Y/[N]) y
   .
   .
   .
   ```

Boot||Shut: Booting a Virtual Partition

To boot a single virtual partition, use either the Monitor command `vparload` or the shell command `vparboot`.

From ISL or EFI>

To boot the existing virtual partition `winona1` from ISL or EFI:

```
ISL> hpux /stand/vpmon vparload -p winona1
```

From MON>

To boot virtual partition `winona1` from the Monitor:

```
MON> vparload -p winona1
```

From HP-UX shell prompt

To boot virtual partition `winona2` from another virtual partition `winona1`:

```
winona1# vparboot -p winona2
```

NOTE

— If the `vparboot` fails but `vparstatus` shows the target virtual partition as down, please try the `vparboot` again after waiting a few seconds. There is a small window of time after a virtual partition is downed by the `shutdown` or `vparreset` command before you can perform the `vparboot` command successfully.

— (PA-RISC only) On nPartitionable servers, memory assigned to a virtual partition is scrubbed as part of the boot process. This will increase boot times, proportional to the amount of memory assigned the virtual partition. Further, if the virtual partition that is being booted owns the hardware console port, there will be a pause in the console output. For more information, see “Switchover Pause with Shutting Down” on page 40.

— When there is a pending reboot for reconfiguration for the involved nPartition, the target virtual partition of the vparload or vparboot commands will not be booted until all the virtual partitions have been shutdown and the vPars Monitor rebooted. For more information see “Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)” on page 135.

— For memory considerations when booting, see “Memory: Allocation Concepts and Notes” on page 178.

Boot||Shut: Shutting Down or Rebooting a Virtual Partition

A virtual partition can be gracefully shut down or rebooted via the HP-UX command `shutdown`. To ensure that the partition database is synchronized (see “vPars Partition Database” on page 32), execute the `vparstatus` command prior to executing the `shutdown` command.

Examples

- To shutdown the virtual partition `winona1`:

  ```
  winona1# vparstatus
  winona1# shutdown -h
  ```

  After `winona1` is shutdown, the virtual partition is in the `down` state. For more information, see “Virtual Partition States” on page 114.

- To reboot the virtual partition `winona1`:

  ```
  winona1# vparstatus
  winona1# shutdown -r
  ```

NOTE

— If a virtual partition has its autoboot attribute set to `MANUAL`, the virtual partition will only halt and will *not* reboot when the command `shutdown -r` (or `reboot -r`) is given. For more information on the virtual partition attributes, see the *vparmodify* (1M) manpage and “Managing: Modifying Attributes of a Virtual Partition” on page 128.

— For the `-R` and `-r` options of the `shutdown` and `reboot` commands, the virtual partition will not reboot when there is a pending RFR (Reboot-for-Reconfig) until all the virtual partitions within the nPartition have been shutdown and the vPars Monitor has been rebooted. Also, the requested reconfiguration will not take place until all the virtual partitions have been shutdown and the vPars Monitor has been rebooted.

— When you need to force a non-graceful shutdown, such as when a partition appears hung, use `vparreset`. See “Resetting a Virtual Partition” on page 152.

— The shell commands `shutdown` and `reboot` apply only to the OS instance of the virtual partition from which they are executed and do not shut down or reboot any other virtual partitions or the vPars Monitor.

— There is no command to shutdown the Monitor. The Monitor command `reboot` (see “Monitor: Using Monitor Commands” on page 105) applies to the entire hard partition, causing the hard partition to reboot. For more information on how to shut down or reboot the hard partition gracefully, see “Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)” on page 135.

When to Shutdown All Virtual Partitions

The only times you need to shutdown all the virtual partitions within a hard partition are when:

- a hardware problem or nPartition modification requires the nPartition to be down. Note that PCI OL* is supported on vPars A.03.xx and A.04.
- the entire hard partition hangs. This *might* be a problem with the Monitor.
- you need to update the vPars Monitor (`/stand/vpmon`)

Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)

To halt or reboot the hard partition gracefully, you need to do the following:

1. Log into *every* virtual partition that is running and gracefully shutdown the partition via the HP-UX command `shutdown`.

   There is no command that shuts down all the virtual partitions at the same time. You need to shutdown every virtual partition one at a time.

   For our example, if all our partitions were up, we would need to shut them down:

```
winona1# vparstatus
winona1# shutdown -h

winona2# vparstatus
winona2# shutdown -h

winona3# vparstatus
winona3# shutdown -h
```

2. After the last virtual partition is shutdown, you will be at the Monitor prompt (MON>) on the console.

   a. To reboot the hard partition, use the Monitor command `reboot`:

```
MON> reboot
```

   b. To shutdown the rp5470/L3000 or rp7400/N4000 servers, access the GSP using `Ctrl-B`. You can then use the GSP command `PC` to power off the server. For example:

```
MON> ^B
GSP> PC
```

   Alternatively, you can power off the rp5470/L3000 or rp7400/N4000 servers via the physical power switch.

   Because no partitions are running and Monitor is running only in memory, shutting down the Monitor this way does not corrupt the server's memory.

   c. To power off the cells assigned to the nPartition, access the GSP using `Ctrl-B`. You can then go to the Command Menu and use the command `PE` to power off the cells. For example:

```
MON> ^B
GSP MAIN MENU:
 CO: Consoles
VFP: Virtual Front Panel
 CM: Command Menu
 CL: Console Logs
 SL: Show chassis Logs
 HE: Help
  X: Exit Connection

GSP> cm
Enter HE to get a list of available commands

GSP:CM> PE
```

```
This command controls power enable to a hardware device.

 B - Cabinet
 C - Cell
 I - IO Chassis
     Select Device: c

 Enter cabinet number: 0
 Enter slot number: 6

 The power state is ON for the Cell in Cabinet 0, Slot 6.
 In what state do you want the power? (ON/OFF) OFF

 GSP:CM>
```

WARNING **Before modifying power settings and for detailed warnings and information on power for nPartitionable servers, please read the section "Powering Cells and IO Chassis On and Off" in the manual *HP System Partitions Guide* available at http://docs.hp.com.**

Boot||Shut: Setboot and System-wide Stable Storage

On a vPars system, the `setboot` command does not read from or write to stable storage. Instead, the `setboot` command reads from and writes to the vPars partition database, affecting only the entries of the virtual partition from which the `setboot` command was run.

For example, if you are logged into `winona2` and execute the command:

```
winona2# setboot -b on
```

this would set the autoboot attribute to `AUTO` for the virtual partition `winona2`.

The actions of setboot on a virtual partition are:

setboot option	vPars effect
`-p`	changes the primary boot path of the virtual partition
`-a`	changes the alternate boot path of the virtual partition
`-s`	sets the autosearch attribute of the virtual partition (pre-A.03.02: no effect)
`-b`	sets the autoboot attribute of the virtual partition
no options	displays the primary boot path, alternate boot path, and autoboot attribute of the virtual partition

NOTE To modify stable storage, you must go to the BCH for PA-RISC or EFI Shell for Integrity systems.

The boot path setting for HAA (High-Availability Alternate) is *not* supported for vPars instances. For more information on HAA, see the *HP System Partitions Guide*.

See also “EFI and Integrity Notes” on page 35 for information on EFI variables and booting.

Boot||Shut: Using Primary and Alternate Boot Paths

You can set the primary and alternate boot paths of a virtual partition by using the HP-UX `setboot` command or the vPars command vparmodify and the `BOOT` and `ALTBOOT` attributes.

NOTE Like many other HP-UX applications, MirrorDisk/UX software is supported. However, vPars does not have a knowledge of the mirror configuration. If your boot disk is mirrored, you may want to explicitly configure the mirror disk as the alternate boot path. Also, on Integrity systems, after creating a mirrored boot disk, you will have to do either a 'setboot -[p|a|t] <path> or vparefiutil -u to be able to boot from the mirror later.

To modify the paths in stable storage, you must go to the BCH for PA-RISC or EFI Shell for Integrity systems.

For Integrity systems, including using the `parmodify` command in `vPars` mode and EFI variables and booting, see also "EFI and Integrity Notes" on page 35.

For more information on how `setboot` works on a vPars server, see "Boot||Shut: Setboot and System-wide Stable Storage" on page 137. For more information on the IO attributes, see *vparresources* (5) manpage.

The boot path setting for HAA (High-Availability Alternate) is not supported for vPars instances. For more information on HAA, see the *HP System Partitions Guide*.

Autoboot and Autosearch Attributes

Beginning with vPars A.03.02, there is a new attribute called autosearch, in addition to the existing autoboot attribute. The autosearch attribute has the value of either `search` or `nosearch` (the default is `nosearch`). See the table below for the results of the combination of possible values. For further information on the attributes, see the *vparcreate* (1M) or *vparmodify* (1M) manpages. For information on setting these attributes, see "Managing: Modifying Attributes of a Virtual Partition" on page 128.

Table 5-5 Boot Attempt Results of the autoboot and autosearch Values

autoboot value	autosearch value	resulting boot attempt
`manual`	`nosearch`	no booting of the target virtual partition is attempted.
`auto`	`nosearch`	only the primary path is attempted
`auto`	`search`	attempt to boot the primary path; if boot fails, attempt to boot the alternate path.
`manual`	`search`	non-nPartitionable servers: no booting is attempted nPartitionable servers: do not attempt to boot the primary path; attempt to boot the alternate path. These actions match the nPartitionable firmware actions. For more information, see the *setboot* (1M) manpage.

Setting the Primary or Alternate Boot Paths

In the examples below, suppose you want the virtual partition `winona2` to have its primary boot disk at `0/8/0/0.5.0` and its alternate boot path at `0/8/0/0.2.0`.

Using setboot

Because `setboot` affects only the virtual partition from which you execute the command, execute these commands from `winona2`.

To set the primary boot path:

```
winona2# setboot -p 0/8/0/0.5.0
```

To set the alternate boot path:

```
winona2# setboot -a 0/8/0/0.2.0
```

Using vparcreate

Within the `vparcreate` command, you can specify the primary or alternate boot paths with the `BOOT` and `ALTBOOT` attributes:

To set the primary boot path:

```
winona1# vparcreate -p winona2 -a io:0.8.0.0.5.0:BOOT
```

To set the alternate boot path:

```
winona1# vparcreate -p winona2 -a io:0.8.0.0.2.0:ALTBOOT
```

Or to set both the primary and alternate boot paths on the same command line:

```
winona1# vparcreate -p winona2 -a io:0.8.0.0.5.0:BOOT -a io:0.8.0.0.2.0:ALTBOOT
```

Using vparmodify

If the virtual partitions are created already, you can specify the primary or alternate boot paths with the `BOOT` and `ALTBOOT` attributes within the `vparmodify` command:

To set the primary boot path:

```
winona1# vparmodify -p winona2 -a io:0.8.0.0.5.0:BOOT
```

To set the alternate boot path:

```
winona1# vparmodify -p winona2 -a io:0.8.0.0.2.0:ALTBOOT
```

Or to set the primary and alternate boots paths on the same command line:

```
winona1# vparmodify -p winona2 -a io:0.8.0.0.5.0:BOOT -a io:0.8.0.0.2.0:ALTBOOT
```

Using Primary and Alternate Paths with nPartitions

The vPars database and the nPartition complex profile are entirely separate. Therefore, a change in the vPars database does not change any complex profile data.

A change in the primary or alternate paths in the vPars database does not change the primary or alternate paths in the complex profile. To change the primary or alternate paths for both a virtual partition and its nPartition, you must change the paths for each separately.

NOTE For an EFI system, the above is true even if you use `parmodify` to change the paths. `parstatus` will show them as set; however, once the system is booted into nPars mode, those changes by `parmodify` are not retained.

Example

Original Status:

Suppose a `vparstatus` output of `keira1` showed the alternate boot path to be `0/0/6/0/0.6.0` (irrelevant output omitted):

```
keira2# vparstatus -p keira1 -v
[Virtual Partition Details]
Name:       keira1
State:      Down
Attributes: Dynamic,Autoboot
.
.
.
[IO Details]
   0.0.6
   0.0.6.0.0.5
   0.0.0
   0.0.4
   0.0.2
   0.0.6.0.0.5.0  BOOT
   0.0.6.0.0.6.0  ALTBOOT
```

and its nPartition showed the nPartition’s alternate path to be `2/0/14/0/0.6.0`:

```
keira2# parstatus -p0 -V
[Partition]
Partition Number       : 0
Partition Name         : npar0
Status                 : active
IP address             : 0.0.0.0
PrimaryBoot Path       : 0/0/6/0/0.5.0
Alternate Boot Path    : 2/0/14/0/0.6.0
HA Alternate Boot Path : 0/0/6/0/0.5.0
.
.
.
```

Changing the Virtual Partition’s Path (vPars Partition Database)

To change `keira1`’s alternate boot path to the boot disk at `0/0/6/0/0.4.0`, run the command:

```
keira2# vparmodify -p keira1 -a io:0.0.6.0.0.4.0:ALTBOOT
```

`vparstatus` now shows:

```
keira2# vparstatus -p keira1 -v
[Virtual Partition Details]
Name:       keira1
State:       Down
Attributes:   Dynamic,Autoboot
Kernel Path:  /stand/vmunix
.
.
.
[IO Details]
   0.0.6
   0.0.6.0.0.5
   0.0.0
   0.0.4
   0.0.2
   0.0.6.0.0.4.0  ALTBOOT
   0.0.6.0.0.5.0  BOOT
   0.0.6.0.0.6.0
```

but note that the *nPartition's* alternate path has *not* changed:

```
# parstatus -p0 -V
[Partition]
Partition Number       : 0
Partition Name         : npar0
Status                 : active
IP address             : 0.0.0.0
PrimaryBoot Path       : 0/0/6/0/0.5.0
Alternate Boot Path    : 2/0/14/0/0.6.0
HA Alternate Boot Path : 0/0/6/0/0.5.0
```

Changing the nPartition's Path (Complex Profile Data)

To change the nPartition's alternate path to `0/0/6/0/0.4.0`, run the command:

```
keira2# parmodify -p0 -t 0/0/6/0/0.4.0
Command succeeded.
```

The nPartition's alternate path has now changed:

```
keira2# parstatus -p0 -V
[Partition]
Partition Number       : 0
Partition Name         : npar0
Status                 : active
IP address             : 0.0.0.0
PrimaryBoot Path       : 0/0/6/0/0.5.0
Alternate Boot Path    : 0/0/6/0/0.4.0
HA Alternate Boot Path : 0/0/6/0/0.5.0
```

Booting Using the Primary or Alternate Boot Paths

To boot `winona2` using the primary path:

```
winona1# vparboot -p winona2 -B pri
```

However, because the primary boot path is the default, you can omit the `-B` portion:

```
winona1# vparboot -p winona2
```

To boot `winona2` using the alternate path:

```
winona2# vparboot -p winona2 -B alt
```

NOTE

- Setting a path using `vparmodify` requires the target virtual partition to be down; `setboot` does not. However, `setboot` can change only the path(s) of the virtual partition from which the `setboot` command is run (in other words, the local virtual partition).
- You cannot specify `pri` or `alt` at the Monitor prompt. However, because the primary boot path is the default, you can boot `winona2` using the primary path using the following command:

  ```
  MON> vparload -p winona2
  ```

  If you want to boot `winona2` using the alternate boot path, you can specify the hardware address for the alternate boot path. For example, to boot the virtual partition `winona2` using the disk at `0/8/0/0.2.0`:

  ```
  MON> vparload -p winona2 -B 0.8.0.0.2.0
  ```

Boot||Shut: Autoboot

The AUTO File on a Virtual Partition

On a non-vPars server, the LIF's `AUTO` file on the boot disk can contain a boot string that includes boot options, such as `-lq` for booting without quorum, or a boot kernel path, such as `/stand/vmunix.other` for booting an alternate kernel (for 11i v2 systems, alternate kernels are in `/stand/alternate_config/`). The `AUTO` file can be changed either through LIF shell commands or `mkboot`.

However, on a vPars server, the LIF's `AUTO` file is read *only* on server bootup; for example, the `AUTO` file might contain "`hpux /stand/vpmon`" (PA-RISC) or "`boot vpmon`" (Integrity), which causes the vPars Monitor to be booted when the server is booted. The `AUTO` file is not read when a virtual partition is booted.

To simulate the `AUTO` file effect when a partition is booted, you can modify the boot options and boot path entries in the vPars partition database via `vparmodify`:

Examples

- On a non-vPars server, to change the `AUTO` file to use the boot options `-lq`, the command is:
  - — PA-RISC: `# mkboot -a "hpux -lq"` *`raw_device_file`*
  - — Integrity: `# mkboot -a "boot vmunix -lq"` *`raw_device_file`*

  On a vPars server, to get the same effect when the partition `winona2` is booted, modify the partition database using `-o` (boot options):

  ```
  # vparmodify -p winona2 -o "-lq"
  ```

- On a non-vPars server, to change the `AUTO` file to use a different kernel, the command is:
  - — PA-RISC: `# mkboot -a "hpux /stand/vmunix.other"`*`raw_device_file`*
  - — Integrity: `# mkboot -a "boot /stand/vmunix.other"` *`raw_device_file`*

  On a vPars server, to get the same effect when the partition `winona2` is booted, modify the partition database using `-b` (boot path):

  ```
  # vparmodify -p winona2 -b "/stand/vmunix.other"
  ```

NOTE For HP-UX 11i v2 (11.23) systems, alternate kernels are in `/stand/alternate_config/`

On a vPars server, the HP-UX command `mkboot` does modify the LIF's `AUTO` file. However, on a vPars server, what is booted initially is the vPars Monitor; then the Monitor boots the virtual partitions. Therefore, what can be in the LIF `AUTO` file is a boot string that boots the Monitor.

Autobooting the vPars Monitor and Virtual Partitions

You can setup the Monitor and all virtual partitions to boot automatically at power up. To do this, make sure the following four conditions are met:

1. **The hard partition's primary and alternate boot paths point to the boot disks of different virtual partitions.**

   For example, to set the primary and alternate boot paths at BCH or EFI:

   ```
   pa pri 0/0/2/0.6.0
   ```

   ```
   pa alt 0/8/0/0.5.0
   ```

2. **The autoboot flag in stable storage is set to `ON`.**

   To set the autoboot flag to `ON` at BCH or EFI:

   ```
   auto on
   ```

3. **The contents of the `AUTO` files of the primary and alternate boot disks contain the boot string for booting the Monitor. The `-a` option of `/stand/vpmon` boots all the virtual partitions that have the autoboot flag set.**

   — PA-RISC: `"hpux /stand/vpmon -a"`
   — Integrity: `"boot vpmon -a"`

   To set the contents of the `AUTO` file on the LIF, log into the virtual partitions that own the primary and alternate boot disks, and execute the `mkboot -a` command:

   For example, after logging into `winona1` which owns the primary boot disk at `0/0/2/0.6.0`, execute:

   — PA-RISC: `winona1# mkboot -a "hpux /stand/vpmon -a" /dev/rdsk/c2t6d0`
   — Integrity: `winona1# mkboot -a "boot vpmon -a" /dev/rdsk/c2t6d0`

   and after logging into `winona2` which owns the alternate boot disk at `0/8/0/0.5.0`, execute:

   — PA-RISC: `winona1# mkboot -a "hpux /stand/vpmon -a" /dev/rdsk/c1t5d0`
   — Integrity: `winona1# mkboot -a "boot vpmon -a" /dev/rdsk/c1t5d0`

4. **The autoboot flag of all the virtual partitions is set to `AUTO`. If applicable *and desired*, set the autosearch flag of all the virtual partitions to `SEARCH`.**

   `AUTO` is the default. However, if you need to reset these values to `AUTO`:

   ```
   winona1# vparmodify -p winona1 -B auto
   winona1# vparmodify -p winona2 -B auto
   winona1# vparmodify -p winona3 -B auto
   ```

   `SEARCH` is *not* the default value. If you wish to set the autosearch attribute to `SEARCH`:

   ```
   winona1# vparmodify -p winona1 -B search
   winona1# vparmodify -p winona2 -B search
   winona1# vparmodify -p winona3 -B search
   ```

NOTE For Superdome and other nPartitionable servers, you must use the boot device path "path flags" to set automatic booting past the BCH for an nPartition. See the manual *HP System Partitions Guide* for more information, including the proper configuration of paths for an nPartition.

When booting multiple virtual partitions automatically, the sequence for booting is not deterministic, and booting a sequence of virtual partitions automatically is not supported. If booting a sequence is required, the sequence of virtual partitions needs to be booted manually (one by one). For more information on booting a virtual partition, see “Boot||Shut: Booting a Virtual Partition” on page 132.

When booting multiple virtual partitions automatically, there is no way to tell which virtual partition will be active with the console after the partitions have booted.

All changes to stable storage can only be performed at the `BCH>` prompt. See “System-wide stable storage and the setboot command” on page 26.

If you need to reboot the hard partition as part of the process to access the `BCH>`, see “Boot||Shut: Shutting Down or Rebooting the nPartition (OR Rebooting the vPars Monitor)” on page 135.

For information on accessing and using the BCH commands, please see your hardware manual.

Boot||Shut: Single-User Mode

It is occasionally necessary to boot HP-UX into single-user mode to diagnose issues with networking or other components. On a non-vPars server, you do this by using the `-is` option at the ISL prompt:

```
ISL> hpux -is
```

On a vPars server, you can boot a virtual partition into single-user mode either at the Monitor prompt or at the HP-UX shell prompt of a running partition.

For example, if we wanted to boot `winona2` into single user mode:

From MON>

From the Monitor prompt, specify the `-is` option as an argument to `vparload`.

```
MON> vparload -p winona2 -o "-is"
```

From HP-UX shell prompt

From the HP-UX shell prompt of another virtual partition, specify the `-o` option with the `vparboot` command:

```
winona1# vparboot -p winona2 -o “-is”
```

NOTE To boot a virtual partition, the partition must be in the `down` state. If the partition is in the `hung` state, perform the following before executing the `vparboot`:

1. Turn off autoboot for the target partition:

   ```
   winona1# vparmodify -p winona2 -B manual
   ```

2. Attempt to reset the target partition with the `-t` option (soft reset):

   ```
   winona1# vparreset -p winona2 -t
   ```

3. If it still appears to be hung, reset it with the `-h` option (hard reset):

   ```
   winona1# vparreset -p winona2 -h
   ```

4. Continue verifying the state until `vparstatus` shows that `winona2` is in the `down` state:

   ```
   winona1# vparstatus -p winona2 -v | grep -E "Name|State"
   Name: winona2
   State: down
   ```

After you have entered into single-user mode and if you want to turn autoboot back on, the command is:

```
winona1# vparmodify -p winona2 -B auto
```

Boot||Shut: Other Boot Modes

In the same way you can boot a virtual partition into single-user mode (see “Boot||Shut: Single-User Mode” on page 146), you can boot a partition using other boot options. The general syntax is:

From MON>

```
MON> vparload -p <target_partition> <boot_options>
```

or

From HP-UX shell prompt

```
<active_partition># vparboot -p <target_partition> -o "<boot_options>"
```

Examples, including using **maintenance mode** with LVM and **overriding quorum** with Mirror-UX, are shown below. For more information on all the boot options, see the manpage *hpux* (1M).

Maintenance Mode

When troubleshooting LVM, you may need to enter into maintenance mode using the `-lm` option. For more information on maintenance mode, see the book “Managing Systems and Workgroups.”

On a non-vPars server, you would boot the server into maintenance mode by executing the following:

```
ISL> hpux -lm
```

On a vPars server, you specify the same `-lm` option but as an argument to either the Monitor `vparload` command or as a `-o` option to the shell `vparboot` command.

For example, if the partition `winona2` is down, to boot `winona2` into maintenance mode:

From MON>

From the Monitor prompt:

```
MON> vparload -p winona2 -o "-lm"
```

From HP-UX shell prompt

From the running partition `winona1`:

```
winona1# vparboot -p winona2 -o "-lm"
```

Overriding Quorum

In LVM, when the root disk is mirrored, the server can only activate the root volume group, which contains the OS instance, when the majority of the physical volumes in a root volume group are present at boot time. This is called establishing a quorum. Sometimes, you may want to boot an OS instance regardless of whether a quorum is established. You can override the quorum requirement by using the `-lq` option. For more information on quorums, see the book "Managing Systems and Workgroups."

On a non-vPars server, you would boot overriding quorum using:

```
hpux -lq
```

On a vPars server, you can execute either of the following:

From MON>

From the Monitor prompt, to boot `winona2` overriding the quorum requirement:

```
MON> vparload -p winona2 -o "-lq"
```

From HP-UX shell prompt

From the running virtual partition `winona1`, to boot `winona2` overriding the quorum requirement:

```
winona1# vparboot -p winona2 -o "-lq"
```

NOTE Specifying the boot options from the command line only affects the current boot.

On a non-vPars server, to have a server permanently boot with the `-lq` option, you would put "`hpux -lq`" (PA-RISC) or "`boot vmunix -lq`" (Integrity) in the LIF `AUTO` file. On a vPars server, to have a partition boot with the `-lq` option, you would simulate the `AUTO` file usage by entering the `-lq` option into the partition database. See “The AUTO File on a Virtual Partition” on page 143.

Changing the LVM Boot Device Hardware Path for a Virtual Partition

Example

Below are the steps to move the root disk of a single virtual partition.

Verification

These instructions require that the virtual partition be constrained
in the following way:

> the logical volume used for the primary swap device must be
> on the boot device; in other words, boot and swap must be on the same disk device.

This can be verified by the following steps:

Step 1. Run lvlnboot.

```
lvlnboot -v /dev/vg00
```

Step 2. Examine the output to identify the “Boot” and “Swap” logical volumes. For example:

```
Boot: lvol1 on: /dev/dsk/c1t6d0
Swap: lvol2 on: /dev/dsk/c1t6d0
```

Step 3. Make sure that the boot and swap logical volumes are on the same device.

CAUTION If the boot and swap logical volumes are not on the same device, do not proceed with these instructions. You will need to contact HP for assistance.

Preparation

Before changing the hardware path of the boot device:

Step 1. Create a mapfile for the root volume group. Keep the mapfile in the root (/) directory, so that it is accessible during single user mode boot.

```
vgexport -p -m /mapfile.vg00 /dev/vg00
```

Step 2. Get a list of physical volumes (PVs) in the root volume group. Keep the PV list file in the root (/) directory, so that it is accessible during single user mode boot.

```
vgexport -p -f /pvs.vg00 /dev/vg00
```

Step 3. You may now shutdown the virtual partition and physically move the disk.

Change the boot device hardware path

Step 1. From another virtual partition, change the target virtual partition attributes

```
vparmodify -p partition_name -a io:new_path:boot -B manual
vparmodify -p partition_name -d io:old_path
```

where

partition_name is the target virtual partition
new_path is the new hardware path of the disk
old_path is the old hardware path of the disk

Step 2. Verify the attributes

```
vparstatus -v -p partition_name
```

Boot into LVM maintenance mode

Step 1. Boot the target virtual partition into LVM maintenance mode. For example, at the Monitor prompt:

```
MON> vparload -o -lm -p partition_name
```

LVM maintenance mode steps

Step 1. Once the partition comes up in LVM maintenance mode, run `ioscan` to get the device filename of the boot device

```
ioscan -fnkCdisk
```

If the device filename (`/dev/dsk/file`) is new, use `insf` to install the special files in `/dev` directory.

Step 2. Run `vgscan` to get the device filenames grouped with the boot device.

```
vgscan
```

Step 3. Remove the old information about root volume group.

```
vgexport /dev/vg00
```

You may have to remove `/etc/lvmtab`.

Step 4. Prepare to import the root volume group (`vg00`).

```
mkdir /dev/vg00
mknod /dev/vg00/group c 64 0x00000
```

Step 5. Import the root volume group (`vg00`). For example:

```
vgimport -m /mapfile.vg00 /dev/vg00 /dev/dsk/c1t1d0 /dev/dsk/c1t1d1
```

where the device filenames are obtained from the `ioscan` and `vgscan` above

Step 6. Activate the root volume group (`vg00`):

```
vgchange -a y /dev/vg00
```

You may also have to cleanup and prepare LVM logical volume to be root, boot, primary swap, or dump volume as follows:

```
lvrmboot -r /dev/vg00
lvlnboot -b /dev/vg00/lvol1
lvlnboot -r /dev/vg00/lvol3
lvlnboot -s /dev/vg00/lvol2
lvlnboot -d /dev/vg00/lvol2
mount
```

Step 7. Verify that the hardware path for the boot device matches the primary boot path.

```
lvlnboot -v /dev/vg00
```

Step 8. If the hardware path has not changed to the primary boot path, change it by running `lvlnboot` with the recovery (`-R`) option. This step is normally not necessary.

```
lvlnboot -R /dev/vg00
```

Step 9. Reboot the target virtual partition.

Resetting a Virtual Partition

Just as it is occasionally necessary to issue a hard reset (RS) or a soft reset (TOC) for a non-vPars OS instance, it is occasionally necessary to issue similar resets for a vPars OS instance.

Hard Reset

On hard partition not running vPars, a hard reset cold boots the hard partition. To issue a hard reset, the administrator types a `CTRL-B` at the console to connect to the service processor and then types the command `RS` (reset), at which time the hard partition cold boots.

On a hard partition running vPars, a hard reset will reset the hard partition--including the Monitor and all the virtual partitions.

To simulate a hard reset on only a virtual partition, from a running virtual partition, use `vparreset` with the `-h` option. For example, if `winona2` is hung, we can execute `vparreset` from the running partition `winona1`:

```
winona1# vparreset -p winona2 -h
```

The `–h` option also inhibits the autoboot behavior (just like `shutdown -h` does); therefore `-h` can be used to break out of a reboot loop. Because `-h` overrides the autoboot setting for that virtual partition, the partition must be manually restarted via `vparboot` (e.g., `winona1# vparboot -p winona2`).

Other virtual partitions are unaffected when one virtual partition is reset.

Soft Reset

On a hard partition not running vPars, a soft reset (TOC) allows HP-UX to attempt to capture a state and potentially create a crash dump and then the hard partition reboots. To issue a soft reset, the administrator types a `CTRL-B` at the console to connect to a service processor and then types the command `TC` (transfer of control).

On a hard partition running vPars, a soft reset will take dumps of all the virtual partitions[1] as well as the Monitor image, and then the hard partition reboots.

To simulate a soft reset on only one virtual partition, from a running partition, use `vparreset` with the `-t` (for TOC) option. For example, if `winona2` is hung, we can execute `vparreset` from the running partition `winona1`:

```
winona1# vparreset -p winona2 -t
```

The target virtual partition either shuts down or reboots according to the setting of the `autoboot` attribute of that virtual partition.

Other virtual partitions are unaffected when one virtual partition is reset.

NOTE Unlike the `RS` and `TC` commands, the `vparreset` command also displays Processor Information Module (PIM) data unless the `-q` option is specified.

On Superdome, when there is a pending reboot for reconfiguration, the target virtual partition will not be rebooted until all the virtual partitions within the nPartition are shut down and the virtual partition Monitor is rebooted.

1. See note titled “Kernel Dumps” on page 233.

Using an Alternate Partition Database File

By default, the local copy of the vPars partition database is kept in the file `/stand/vpdb` on the boot disk of each virtual partition within a hard partition. However, you can create, edit, and delete virtual partitions in an alternate partition database file by using the "`-D` *`filename`*" option in the vPars command string, where *`filename`* is the name of the alternate partition database file. For more information on the vPars command strings, see the vPars manpages.

The alternate partition database file can be used to create an entirely different virtual partition configuration without affecting the live partition database in the Monitor's memory or the local copies in `/stand/vpdb`.

Example

Suppose the current virtual partition configuration is:

Partition Name	winona1	winona2	winona3

Bound CPUs	`total = 2`
	`min = 2`
		`total = 2`
		`min = 2`
		`paths = 41,45`
			`total = 1`
			`min = 1`

Unbound CPUs	`three CPUs are available`		
Memory	`1024 MB`	`1280 MB`	`1280 MB`

IO Paths (LBAs)	`0.0`
	`0.4`
		`0.8`
		`1.10`
			`0.5`
			`1.4`

Boot Path	`0.0.2.0.6.0`	`0.8.0.0.5.0`	`1.4.0.0.5.0`
LAN	`0.0.0.0`	`1.10.0.0.4.0`	`0.5.0.0.4.0`
Autoboot	`AUTO`	`AUTO`	`AUTO`

You could create an alternate partition database where the configuration is:

Partition Name	winsim1	winsim2

Bound CPUs	`total = 4`
	`min = 4`
		`total = 4`
		`min = 4`

Unbound CPUs	`no CPUs are available`	
Memory	`1600 MB`	`1600 MB`

IO Paths (LBAs)	`0.0`
	`0.4`
		`0.8`
		`1.10`
		`1.2`

Boot Path	`0.0.2.0.6.0`	`0.8.0.0.5.0`
LAN	`0.0.0.0`	`1.10.0.0.4.0`

Autoboot	AUTO	AUTO

To create and boot using an alternate partition database, perform the following:

1. Create the partition configuration and alternate partition database file.

```
winona1# vparcreate -p winsim1 -D /stand/vpdb.sim -a cpu::4 -a cpu:::4 -a mem::1600 -a
io:0.0 -a io:0.4 -a io:0.0.2.0.6.0:BOOT
winona2# vparcreate -p winsim2 -D /stand/vpdb.sim -a cpu::4 -a cpu:::4 -a mem::1600 -a
io:0.8 -a io:1.10 -a io: 1.2 -a io:0.8.0.0.5.0:BOOT
```

CAUTION *LBAs must be explicitly specified*. The examples in this chapter use a non-nPartitionable system. If using an nPartitionable system, please read "Planning, Installing, and Using vPars with an nPartitionable Server" on page 48.

The alternate partition database file is created if it does not exist.

NOTE In order to boot from an alternate partition database file, the file must exist in /stand of the disk from which you will boot the entire server.

2. Shutdown all the virtual partitions and reboot the server:

```
winona3# vparstatus ; shutdown -hy 0
winona2# vparstatus ; shutdown -hy 0
winona1# vparstatus ; shutdown -hy 0
MON> reboot
```

3. Interrupt the boot process and boot the Monitor /stand/vpmon specifying the -p alternate partition database option and the -a autoboot option:

```
BCH> bo pri
interact with IPL: y

ISL> hpux /stand/vpmon -D /stand/vpdb.sim -a
```

The Monitor boots, reads the partition database file /stand/vpdb.sim, and copies the partition configuration information into the Monitor's memory. The local copy of the partition database is now /stand/vpdb.sim (the same filename as what was read by the Monitor at Monitor boot time).

Integrity NOTE: If you issue `readdb /stand/vpdb.backup`, the file that is actually read is at /stand/boot.sys/stand/vpdb.backup. The `vparcreate` command transparently creates the soft link from /stand/boot.sys/stand/<file> to /stand/<file>. Therefore, if you backup the database file using the Unix `cp` command, a `ln` command also should be executed to create the soft link. Otherwise it will not be possible to boot from the backup database file.

Because the local copy is now /stand/vpdb.sim, you do not need to specify the `-D /stand/vpdb.sim` option when performing vPars Monitor commands. For example, to set the `static` attribute for the partition `winsim2`, the command is:

```
winsim2# vparmodify -p winsim2 -S static
```

This change will be synchronized to the local copies of /`stand/vpdb.sim`. (If /stand/vpdb.sim does not exist, as in this case on `winsim2`, the file will be automatically created during synchronization).

4. To return to using `/stand/vpdb`, do the same steps as above, except on the ISL command line in Step 3 is:

   ```
   ISL> hpux /stand/vpmon -a
   ```

   By default, the file `/stand/vpdb` is read as the partition database file.

When working with an alternate partition database file using `-D` *`filename`*, please note the following:

- *`filename`* must reside in `/stand` when the server boots because the vPars Monitor can only traverse HFS file systems of the boot disk.
- Be careful when creating partitions using the `-D` option. Fewer checks on configuration are being performed. It is possible to create a partition configuration that is not valid.
- All LVM rules still apply. For example, you cannot migrate IO only by re-assigning the IO to a different partition; you must still `vgexport` and `vgimport` the volume groups.
- (pre A.03.02) Although there is no command that displays which partition database file was read when the Monitor was booted, because the local copies of the active database are synchronized every five seconds, you should be able to tell which database file was read and is active based on the time stamps of the various database files in /`stand`.

6 CPU, Memory, and IO Resources (A.04.xx)

Managing Hardware Resources

- IO Allocation (Adding or Deleting IO Resources)
- Memory Allocation (Firmware Configuration and Adding or Deleting Memory Resources)
- CPU Allocation (Adding or Deleting CPU Resources)
- Using iCAP (formerly known as iCOD) with vPars
- CPU Monitor (deallocation and deconfiguration)

NOTE Some examples in this chapter may use a non-nPartitionable system where there is no cell in the hardware path. If using an nPartitionable system you must include the cell in the hardware path; please read “Planning, Installing, and Using vPars with an nPartitionable Server” on page 48.

IO: Concepts

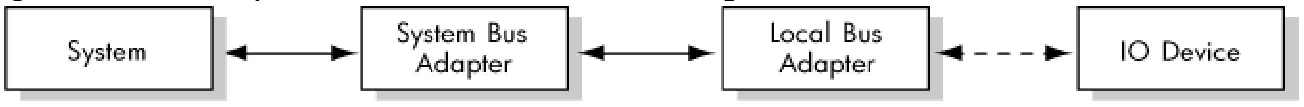
Acronyms

LBA	Local Bus Adapter
SBA	System Bus Adapter

System, Cells, SBA, LBA, Devices and Relationships

On a server, an IO device communicates to the system through the LBA and SBA. The path looks like

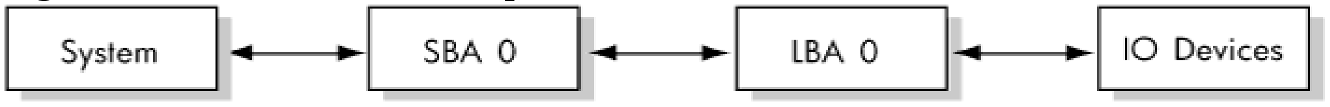
Figure 6-1 System to IO Device Relationship



This corresponds to the `ioscan` hardware path output for an IO device of `sba/lba/ ... /device`.

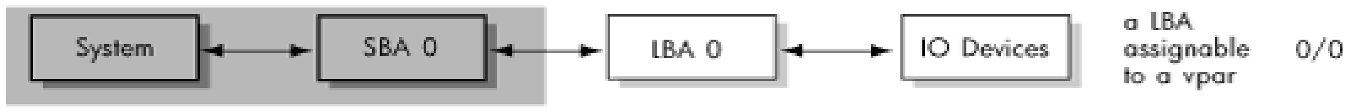
A LBA actually owns all the devices attached to it. In the example below, all the IO devices attached to LBA 0 are owned by LBA 0, and the hardware paths of those IO devices begin with 0/0 (sba/lba). (Cells are discussed later and would change the hardware path to cell_ID/sba/lba.)

Figure 6-2 LBA owns Multiple IO Devices



It is at the LBA level where vPars assigns IO. In the example below, this means that LBA 0 can be assigned to at most one virtual partition. If LBA 0 is assigned to vparN, it is implied that all IO devices attached to LBA 0 are assigned to vparN.

Figure 6-3 vPars allocates IO at the LBA Level

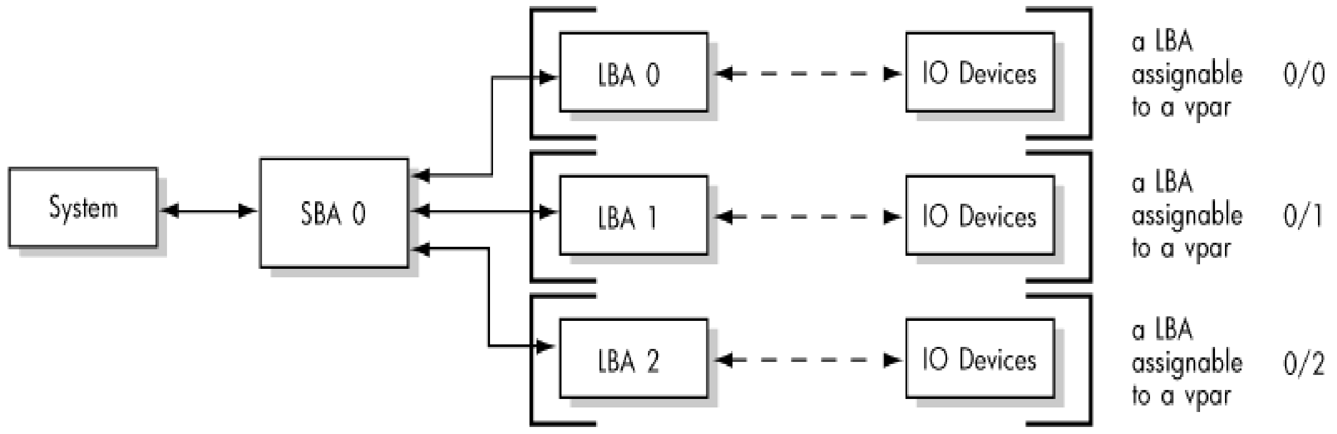


A SBA has multiple LBAs attached to it; it is a hierarchical relationship. Nevertheless, assignments in vPars remain at the LBA level, and each LBA can be assigned to a different virtual partition.

NOTE Regarding syntax and how vPars commands interpret what is specified on the command line, see "IO: Allocation Notes" on page 162. Even if there are shortcuts in assigning LBAs, vPars assigns per LBA.

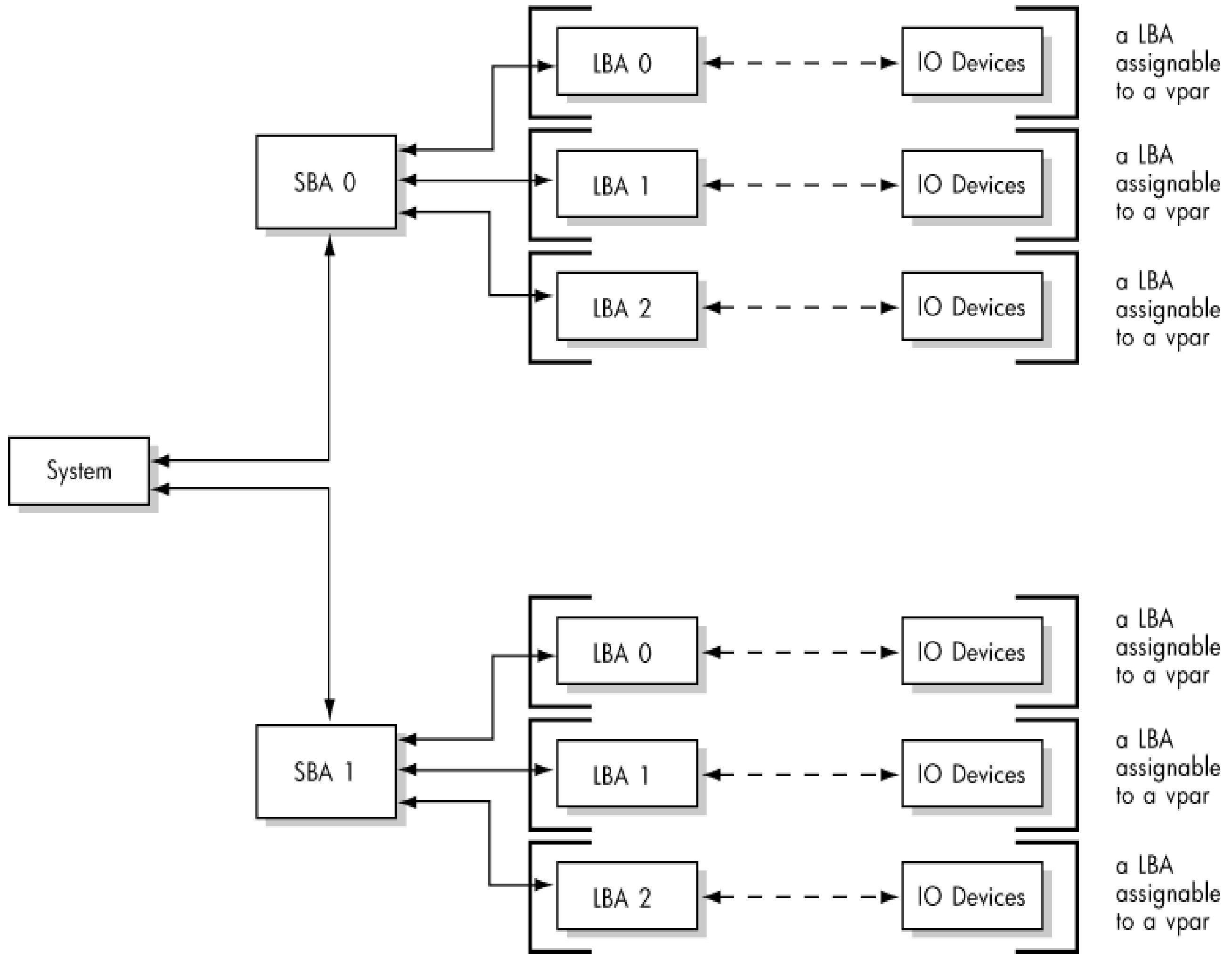
In the example below, each LBA (shown in brackets) can be assigned to a different virtual partition.

Figure 6-4 vPars allocates at LBA level not SBA level


System
SBA 0
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
0/0
a LBA assignable to a vpar
0/1
a LBA assignable to a vpar
0/2

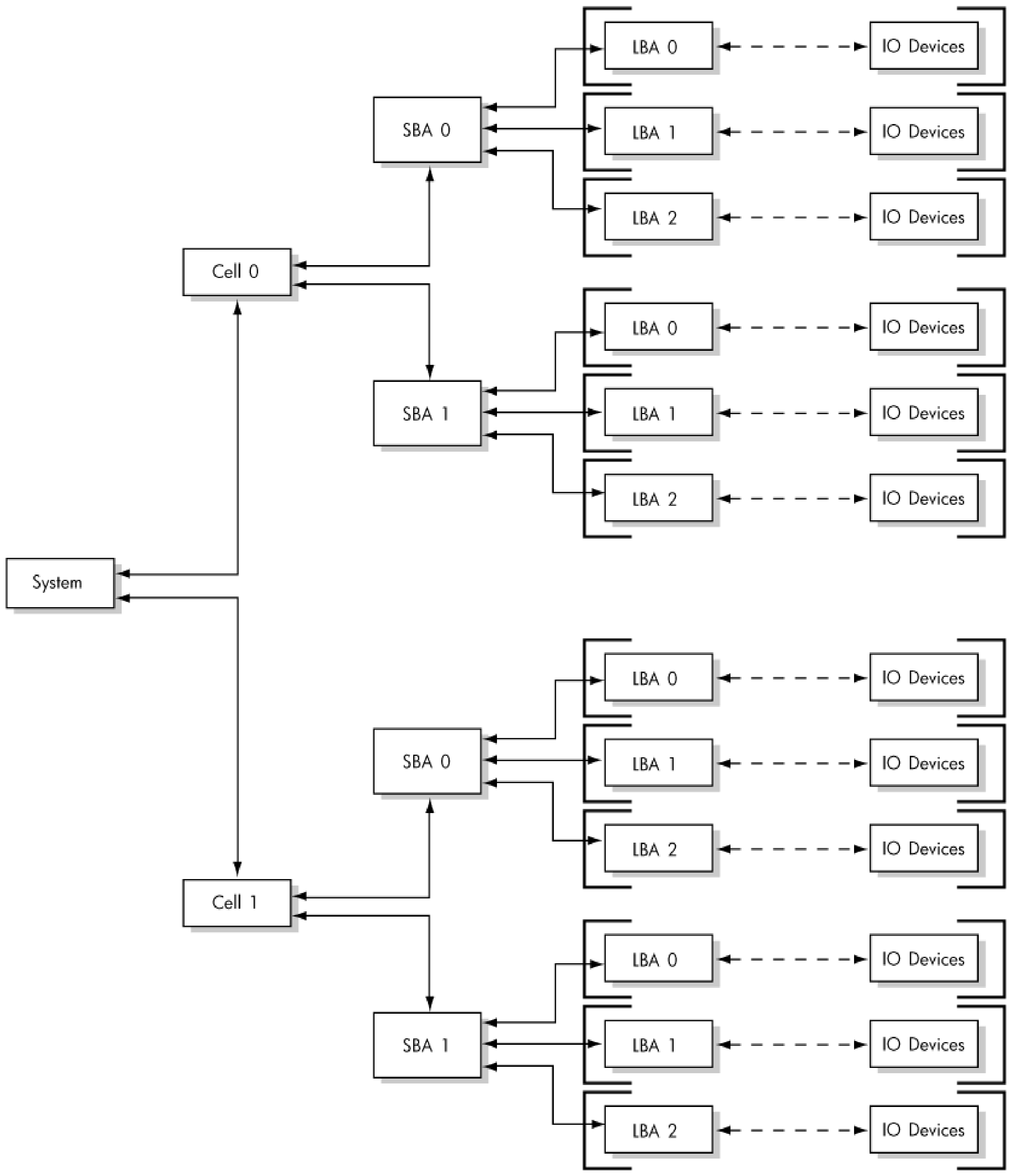
A system has multiple SBAs, but assignments remain at the LBA levels.

Figure 6-5 vPars allocates at LBA level not SBA level


SBA 0
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
a LBA assignable to a vpar
a LBA assignable to a vpar
System
SBA 1
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
a LBA assignable to a vpar
a LBA assignable to a vpar

With the addition of cells (an nPartitionable server), there are more SBAs, but IO assignments remain at the LBA level:

Figure 6-6 vPars allocates at LBA level not at cell level



IO: Adding or Deleting LBAs

IO Syntax in Brief

the basic core syntax for adding or deleting IO resources is

`-a|d io:`*`hardware_path`*

where

`a`	is adding
`d`	is deleting
`hardware_path`	is the hardware path of the IO

Examples

- To add all hardware using the SBA/LBA hardware path of `1/2` to an existing partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -a io:1.2
  ```

- To remove all hardware with SBA/LBA `1/2` from partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -d io:1.2
  ```

NOTE The virtual partition must be in the `down` state to add or delete IO resources.

IO: Allocation Notes

When planning or performing IO allocation, note the following:

- **An LBA can be assigned to at most one virtual partition at any given time.**

  When you are planning your IO to virtual partition assignments, note that only one virtual partition may own any hardware at or below the LBA (Local Bus Adapter) level. In other words, **hardware at or below the LBA level must be in the same virtual partition**.

  Example

  Looking at the `ioscan` output of a rp7400/N4000, the two internal disk slots use the same LBA:

```
0/0        ba              Local PCI Bus Adapter (782)
0/0/2/0        ext_bus     SCSI C875 Ultra Wide Single-Ended
0/0/2/1        ext_bus     SCSI C875 Ultra Wide Single-Ended
```

  Therefore, you *cannot* assign one of the internal disks to partition `vpar1` and the other internal disk to partition `vpar2`; these disks must reside in the same partition.

- **Syntax Notes**

  When specifying only the SBA on the command-line, the vPars commands will assume the change applies to all LBAs under the specified SBA.

  The exception are boot disks; **boot disks are specified using the full hardware path**.

NOTE When assigning IO, if you specify a path below the LBA level (for example, `cell/sba/lba/.../device`, vPars automatically assign the LBA to the virtual partition. For example, if you specify `-a io:0/0/0/2/0.6.0` where `0/0/0` is the cell/sba/lba, the lba of 0/0/0 is assigned to the virtual partition. Further, this LBA assignment implies that all devices using `0/0/0` are assigned to the virtual partition.

The assignment rules of LBAs remain applicable: the LBA can only be owned by one virtual partition. For example, once the LBA at 0/0/0 is assigned to one virtual partition, it cannot be simultaneously assigned to any other virtual partition. Thus, if the device at `0/0/0/2/0.6.0` is assigned to a virtual partition, the LBA at 0/0/0 is assigned to that virtual partition, so the device at `0/0/0/2/0.6.0` cannot be assigned to a different virtual partition.

LBA Example

The `vparcreate` command on a non-nPartitionable system looks like:

```
#vparcreate -p vpar1 -a cpu::1 -a cpu:::1 -a mem::1024 -a io:0.0 -a io:0.0.2.0.6.0:BOOT
```

where the IO assignment is specified using the LBA level (`-a io:0.0`) and the boot disk is specified using the full hardware path (`-a io:0.0.2.0.6.0`).

For an nPartitionable system, the `vparcreate` command would look like:

```
#vparcreate -p vpar1 -a cpu::1 -a cpu:::1 -a mem::1024 -a io:0.0.0 \
-a io:0.0.0.2.0.6.0:BOOT
```

where the IO assignment is specified using the LBA level (`-a io:0.0.0.`) and the boot disk is specified using the full hardware path (`-a io:0.0.0.2.0.6.0`).

For information on using the LBA level on nPartitionable systems, also see “Planning, Installing, and Using vPars with an nPartitionable Server” on page 48.

- **SBA/LBA versus cell/SBA/LBA**

  When viewing hardware paths, note the following:

  1. The explicit specification of an LBA on a non-nPartitionable system consists of two fields: `sba/lba`
  2. The explicit specification of an LBA on an nPartitionable system consists of three fields: `cell/sba/lba`
  3. With A.04.xx, all LBAs under a SBA are implied when explicitly specifying a SBA without specifying any LBA. Therefore, the path specified on a command line can have different meanings depending upon the vPars version, the type of server, and the user intent. For example, the path of `x/y` can mean *any* of the following:

     — sba=x, lba=y on a non-nPartitionable server running vPars A.03.01 or earlier

     — sba=x, lba=y on a non-nPartitionable server running vPars A.03.02 or later or A.04.xx

     — cell=x, sba=y on an nPartitionable server running vPars A.03.02 or later or A.04.xx

- **Supported IO**

  Check your hardware manual to verify that your mass storage unit can be used as a bootable device. If a mass storage unit cannot be used as a boot disk on a non-vPars server, it cannot be used as a boot disk on a vPars server. vPars does not add any additional capability to the hardware.

  For information on supported IO interface cards and configurations, see the document *HP-UX Virtual Partitions Ordering and Configuration Guide.*

Memory: Concepts and Functionality

Acronyms

ILM	Interleaved Memory. • The nPartition's system default is to have all memory configured as ILM. • vPars A.03.xx and A.02.xx use and assign only ILM; vPars A.04.xx allows use of ILM and CLM.
CLM	Cell Local Memory. • Using nPartition commands, you can re-configure a portion of a cell's ILM memory to be used instead as CLM. Beginning with vPars A.04, you can assign an amount of available CLM to a virtual partition. • This capability should be used only if you are an advanced administrator. Further, CLM is meant to be used in conjunction with CLP (cell local processor); not doing so may actually cause performance degradation.

Assignments

You assign memory to a virtual partition:

- by size

  this uses the nPartition's ILM.
- by cell and a corresponding size

  this uses the specified cell's CLM.

Within the available nPartition's ILM or cell's CLM, you can also

- specify an address range to use

  This does not increase the amount of memory assigned to the virtual partition. The address range is a specific subset of the existing ILM or CLM amount assigned to the virtual partition. Therefore, the total amount of memory specified by ILM or CLM addresses cannot exceed the amount of ILM and CLM assigned to the virtual partition.

NOTE The virtual partition must be in the `down` state to add or delete memory resources.

Granularity

Granularity refers to the unit size in which memory is assigned to the all virtual partitions in a given vPars database (vpdb). You should be careful when using the granularity option; using the granularity option incorrectly can cause all the virtual partitions to not be bootable. For information, see "Memory: Setting Granularity Values" on page 172.

Memory: Assigning By Size (ILM)

Assigning memory by specifying only size uses ILM memory. ILM memory is the only type of memory used in vPars A.03.xx and earlier. vPars A.04.xx and later can use either ILM and CLM memory.

Syntax

The basic syntax for adding or decreasing ILM resources assigned to a virtual partition is

```
-[a|d] mem::size
```

where

a	is adding
d	is decreasing
size	is the quantity of ILM in MBs

Examples

- To create the virtual partition `winona2` with 1024 MB of ILM:

  ```
  winona1# vparcreate -p winona2 -a mem::1024
  ```

- To add 1024 MB of ILM to an existing partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -a mem::1024
  ```

- To decrease the amount of ILM assigned to partition `winona2` by 1024 MB:

  ```
  winona1# vparmodify -p winona2 -d mem::1024
  ```

NOTE Although not an error, the size of ILM assigned should be a multiple of the granularity value. See also “Memory: Assigning CLM” on page 169.

Memory: Configuring CLM for an nPartition in Brief

ILM memory can be re-configured to be CLM using the `parmodify` command. Then, you can assign existing available CLM to a virtual partition.

For complete information on CLM and configuring cells and nPartitions, see the *nPartition's Guide*.

`parmodify` syntax

The syntax for the `parmodify` command is

```
parmodify -p nPar_num -mcell:[type]:[use]:[fail][:clm]
```

where

nPar_num	is the nPartition number
cell	is the cell to be modified. this can be either the global format (`cell_number`) or hardware format (*cabinet*/*slot*)
type	is the cell type. `base` is the only valid value and is the default.
use	is the use-on-next-boot value. this is either a `y` or `n`. The default is `y`. Use `y` if the cell is to be an active member of the nPartition or `n` if the cell is to be an inactive member
fail	is the cell failure usage. the only valid value is `ri` (reactivate with interleave) and is the default
clm	is the amount of memory that will be configured as cell local memory for the cell. this value can either be an absolute (the default) or percentage value. The absolute value is a number, which represents the amount of memory in **gigabytes**; the percentage value is any number followed by the character %.

Example

- To configure 1 GB of memory as CLM on cell 6 of nPartition 0, the command is

  ```
  # parmodify -p0 -m 6::::1
  ```

  For the `parmodify` change to take effect, you need to reboot the nPartition. This can be done by `MON> reboot`.

Configuring CLM Notes

- It is possible that a server's NVRAM (non-volatile RAM) values for ILM and CLM can change due to a firmware update or a previous systems administrator. Please verify the NVRAM values.
- Any memory that is not configured as CLM remains ILM.
- You need to be sure to leave enough ILM memory for your virtual partitions to boot, as specified in your vPars database. Also, on PA-RISC not all memory can be configured as CLM in an nPartition. You need to allow ILM memory for the vPars Monitor to boot and for all the OS kernels to boot; excluding the first granule, a portion of which is used by the Monitor, there must be at least one entire granule that exists below the 2 GB limit for each virtual partition. These granules below the 2 GB limit are used by kernel of each virtual partition.
- nPartition configuration rules limit the minimum amount of configured CLM memory for a cell. If you have any CLM on a cell, it must be at least 512MB.

- When performing parmodify commands within a vPars environment, you will see only the cells that are within the nPartition running vPars. If you wish to add cells from outside the nPartition, you will need to be in standalone (PA-RISC) or nPars (Integrity) mode to see the other cells.

CAUTION

- If you reduce the ILM amount below that configured for all the virtual partitions in your vPars database, one or more of the virtual partitions may not boot. Please be sure that there is enough CLM and ILM configured in the nPartition so that the sum of ILM and CLM memory of all your virtual partitions does not exceed the ILM and CLM in the nPartition.
- New servers *should* be set with 100% ILM and 0% CLM. However, a firmware update or even another systems administrator can change these values. You should check your NVRAM configuration before configuring vPars. Also, if you need to set an equal amount of ILM across all cells (see the HP-UX Virtual Partitions Release Notes for A.04.01), you should set your NVRAM configuration to 100% ILM and 0% CLM, then perform the CLM calculations and operations to get the equal amounts of ILM.

  To check the existing ILM/CLM configuration, use `parstatus`. For example, to check the ILM and CLM configuration in nPartition 2:

```
keira# parstatus -V -p 2
[Partition]
Partition Number       : 2
Partition Name         : Partition 2 - HPUX
Status    : Active
IP Address:
Primary Boot Path      : 12/0/3/1/0.0.0
Alternate Boot Path    :
HA Alternate Boot Path :
PDC Revision           : 3.66
IODCH Version          : ffff
Cell Architecture      : Itanium(R)-based
CPU Compatibility      : BCF-640
CPU Speed : 1600 MHz
Core Cell : cab1,cell4
Core Cell Choice [0]   : cab1,cell4
Total Good Memory Size : 40.0 GB
Total Interleave Memory: 1.0 GB
Total Requested CLM    : 40.0 GB
Total Allocated CLM    : 39.0 GB
```

  The `parstatus` output shows the current configuration of 1 GB ILM and 39 GBs of CLM.

  Assuming nPartition 2 consists of cells 12 and 14, to change values of ILM to 100% and CLM to 0%, set the value of CLM to 0% (any memory not configured as CLM becomes ILM):

```
keira# parmodify -p 2 -m 12::::0% -m 14::::0%
keira# shutdown -R
```

  After the nPartition is rebooted, the `parstatus` output should show all memory configured as ILM:

```
keira# parstatus -V -p 2
[Partition]
Partition Number       : 2
Partition Name         : Partition 2 - HPUX
Status    : Active
IP Address:
Primary Boot Path      : 12/0/3/1/0.0.0
```

```
Alternate Boot Path    :
HA Alternate Boot Path :
PDC Revision           : 3.66
IODCH Version          : ffff
Cell Architecture      : Itanium(R)-based
CPU Compatibility      : BCF-640
CPU Speed : 1600 MHz
Core Cell : cab1,cell4
Core Cell Choice [0]   : cab1,cell4
Total Good Memory Size : 40.0 GB
Total Interleave Memory: 40.0 GB
Total Requested CLM    : 0.0 GB
Total Allocated CLM    : 0.0 G
```

Memory: Assigning CLM

Once CLM is configured (see “Memory: Configuring CLM for an nPartition in Brief” on page 166, the syntax to assign an amount of CLM is

`-a|d cell:`*`cell_ID`*`:mem::`*`size`*

where

`a`	is adding
`d`	is decreasing
`cell_ID`	is the cell number
`size`	is the quantity of memory in MBs

Example

- To add 1024 MB of memory from cell 6 (at least 1024 MB must already be configured as CLM) to the existing partition keira2:

  ```
  keira1# vparmodify -p keira2 -a cell:6:mem::1024
  ```

- You can set both ILM and CLM memory to the same partition. To assign 1024 MB of available CLM and 1024 MB of available ILM to keira2:

  ```
  keira1# vparmodify -p keira2 -a cell:6:mem::1024 -a mem::1024
  ```

NOTE Assigning ILM or CLM memory by size only reserves the *amount* of physical memory the virtual partition gets. The exact physical ranges of memory the virtual partition gets is decided by the Monitor when the virtual partition boots. The Monitor will attempt to pick the memory ranges such that the sum of the ranges add up to the amount of ILM and CLM reserved for the partition. However, due to memory fragmentation, which occurs due to memory already taken by the Monitor, firmware, or bad pages, the sum of the ranges picked by the Monitor may be slightly less than or more than the specified amount reserved for the partition.

Memory: Specifying Address Range

Within the already allocated memory sizes, you can specify the memory address ranges using the `mem:::`*`base`*`:`*`range`* syntax. However, this is not recommended unless you are familiar with using memory addresses and for PA-RISC systems, you should also be familiar with the 2 GB memory requirement for the HP-UX kernel and know the number of virtual partitions you will create.

For usage information, see the *vparmodify* (1M) manpage. You should select your `base:range` after consulting `vparstatus -A` to determine which ranges are available.

NOTE

- Specifying an address range does not increase the amount of memory assigned to the partition. Rather, it only specifies addresses to use for the already allocated memory sizes.

  Therefore, all specified ranges cannot exceed the total allocated memory for the virtual partition. In other words, the sum of the ILM- or CLM-specified ranges cannot exceed the total amount of ILM or CLM memory reserved for the virtual partition.
- (vPars A.04) Address ranges are unique within a given virtual partition. Therefore, specifying `base:range` (and not a cell_ID) is sufficient for using an address range within CLM. You can use `vparstatus -A` to list the available ranges and whether the ranges are a part of ILM or CLM. Further, if the range is within CLM, `vparstatus -A` also lists to which cell the range belongs.

CAUTION Normally, ranges are granule-aligned (in other words, the starting address and the ending address of the range is a multiple of a granule). However, due to memory fragmentation, some of the ranges may not be granule-aligned. It is not recommended to assign such nongranule-aligned ranges as user-specified ranges (`mem:::`*`base`*`:`*`range`*). Further, future vPars releases may not support specifying ranges that are not aligned to a granule and may return an error when such ranges are assigned to a virtual partition.

2 GB Restriction (PA-RISC only)

When ranges are specified for the entire memory owned by a partition, you should ensure that at least one of the ranges is below 2 GB and is large enough to accommodate the kernel for that partition. However, other partitions also require memory below 2 GB for their kernels. Hence, you also should ensure that the specified range below 2 GB is not so large such as to preclude memory below 2 GB for the other partitions.

In general terms, the sum of the size of the kernels must be < 2 GB. To calculate the kernel sizes, see “Calculating the Size of Kernels in Memory (PA-RISC only)” on page 271.

CAUTION Not allowing enough memory for the other partitions will cause the other partitions to not boot. You can boot the partition by freeing up enough memory for the partition to boot, such as by shutting down an active partition.

If no memory ranges are below 2 GBs for a given partition, the partition will not boot.

If you use the defaults of the dynamic tunables, you will not run into the 2 GB limit. However, if you have adjusted the dynamic tunables, it is possible to run beyond the 2 GB boundary. For more information on adjusting the kernel size with dynamic tunables, see the white paper *Dynamically Tunable Kernel Parameters* at http://docs.hp.com.

Memory: Setting Granularity Values

Granularity refers to the unit size in which memory is assigned to all virtual partitions in a given vPars database (vpdb). Granularity reflects only the unit size of memory and not the amount of memory that is assigned.

If you are using a vPars database created using vPars A.04.xx and later, the default granularity is 128 MB for ILM and 128 MB for CLM.

However, you can specify your own granularity for CLM and/or ILM. Granularity has some specific restrictions and cannot be changed in a vPars database after they are set. Please be sure to read the **CAUTION** portion at the end of this section.

Syntax

The syntax for setting granularity unit size is

`-g ILM|CLM:`*`unit`*`[:y|n]`

where

`g`	is granularity
`ILM|CLM`	specifies whether the unit size is applied to `ILM` or `CLM`
`unit`	is the granularity unit size in MBs This value must be an integral power of 2 (in other words, 2^X) and be greater than or equal to 64.
`y|n`	[`vparcreate` only; Integrity only] specifies whether granularity unit size should be written to firmware. the default is `n`.

Granularity Values Locations

Integrity Systems. There are two areas where granularity values are set:

1. nPartition firmware
   (specifically, the EFI variables in NVRAM (non-volatile RAM))
2. vPars database

In order for the virtual partitions in the vPars database to be able to boot, the granularity values in the vPars database must match the granularity values in the firmware. This is true for Integrity only; for PA-RISC, there are no granularity values in the PA-RISC firmware.

On Integrity systems, memory is divided into the granules by the firmware; therefore, it is required that you set and match the corresponding EFI variables.

PA-RISC systems. There is only one area where granularity values are set:

1. vPars database

For PA-RISC, there are no granularity values in the PA-RISC firmware. The memory is divided into the granules by the Monitor itself. Note that this means the update firmware option (`[:y]`) of `vparcreate` is ignored on PA-RISC.

Setting the Granularity Values on Integrity

There are two commands that can set the granularity values. Both are available at the HP-UX shell level and use the `-g` option:

1. `vparenv -g ...`
   writes the granularity values to the firmware only. Note that `vparenv` is applicable only on Integrity.
2. `vparcreate -g ...`
   writes the granularity values to the vPars database and also can (using the update firmware option [:y]) write these values to firmware.

vparenv

```
# vparenv -g ILM|CLM:unit
```

writes the *`unit`* granularity value to firmware. This takes effect for the nPartition on the next nPartition boot. For example, to set the granularity unit size in firmware for ILM to 512 MB and for CLM to 256MB:

```
# vparenv -g ILM:512 -g CLM:256
```

Note that this does not set the granularity value in the vPars database. Only `vparcreate` sets the granularity value in the vPars database.

Because the granularity values in firmware must match those in a vPars database in order for the virtual partitions of that database can boot, once the granularity values in firmware are set, the granularity values in the vPars database that are used in the next boot of the vPars Monitor must contain the same granularity values.

vparcreate

```
# vparcreate -p vpar1_name [ -g ILM:unit[:y] ] [ -g CLM:unit[:y] ]
```

If you specify the above command *without* the `:y`, `vparcreate` writes the *`unit`* granularity value to only the vPars database; it does not write the value to firmware.

If you specify the above command *with* the `:y`, `vparcreate` writes the *`unit`* granularity value to both the vPars database and to firmware.

When using this method, note that the `-g` option must be performed when creating the vPars database (in other words, when performing the initial `vparcreate` command). If you set the value incorrectly using the initial `vparcreate` command, you cannot adjust it later. You must re-create the vPars database.

Usage Scenarios

vparcreate with the :y option

The following scenario is where you would want to use `vparcreate` with the `-g` option and the `:y` specification:

1. In nPars mode, you create your first virtual partition with a 256 MB granularity value for ILM. The command is

   ```
   # vparcreate -g ILM:256:y -p keira1 ...
   ```

2. This writes the ILM granularity value to both the vPars database and to firmware. Since the default CLM granularity value is 128, this also writes the CLM granularity value of 128 to both the vPars database and to firmware. Because the values in both the vPars database and firmware match, you can boot this vPars database immediately after setting the nPartition for `vPars` mode and rebooting the nPartition.

   ```
   # vparenv -m vPars     /* to set the mode */
   ```

```
# shutdown -r            /* reboot the system */
```

`vparcreate` without the :y option and `vparenv`

The following scenario is where you would want to use `vparcreate` with the `-g` option but without the `:y` specification. It also shows where you need to use `vparenv` to set the granularity value in the firmware. Note that this scenario would only occur on Integrity systems.

1. You are in a vPars environment, running the default vPars database of `/stand/vpdb` that uses the 128 MB granularity values for ILM and CLM. Because the virtual partitions have been booted successfully, this means that the current firmware has granularity values of 128MB.
2. You create an alternate database `/stand/vpdb.alt` with a granularity value of 512 MB for ILM and 256 MB for CLM.

   ```
   # vparcreate -D /stand/vpdb.alt -g ILM:512 -g CLM:256 -p keira1 ...
   ```
3. This writes the granularity value to the vPars database but not to firmware, which allows you to continue using the active vPars database `/stand/vpdb` with its 128 MB granularity value.

   When you wish to load `/stand/vpdb.alt`, then you can set the granularity value in firmware using `vparenv`, reboot the nPartition, and load the alternate database.

   ```
   # vparenv -g ILM:512 -g CLM:256          /* to set granularity value in firmware */
   MON> reboot                             /* reboot the nPartition */
   ...
   HPUX> boot vpmon -D /stand/vpdb.alt      /* load the alternate database */
   ```

Setting the Granularity Values on PA-RISC

There is only one command that can set the granularity values: the `vparcreate` command with the `-g` option:

1. `vparcreate -g ...`

`vparcreate` writes the granularity values to the vPars database; the update firmware option ([:y]) is ignored on PA-RISC.

As on Integrity, when using the `vparcreate` command to set the granularity value, you set the granularity values using the initial `vparcreate` command (the first `vparcreate` command creates the vPars database). Because you cannot modify the granularity in the vPars database after it is set, you must set it correctly in your initial `vparcreate` command. Otherwise, to correct an incorrect value, you need to re-create the vPars database.

Usage Scenario

In standalone mode, you create your first virtual partition with a 256 MB granularity value for both ILM and CLM. The command is:

```
# vparcreate -g ILM:256 -g CLM:256 -p keira1 ...
```

This writes the granularity values to the vPars database.

Note that with the vparcreate command, when you specify the granularity value for only one type of memory (ILM or CLM), the granularity value for the other type of memory is set using the default granularity value. For example, if you specify only `-g ILM:256`, the `-g CLM:128` is implied where `128` is the vPars default granularity value.

Granularity Cautions (Integrity and PA-RISC)

CAUTION (`vparcreate` only) When you specify the granularity value for only one type of memory (ILM or CLM), the granularity value for the other type of memory is set using the default granularity value. For example, if you specify only `-g ILM:256`, the `-g CLM:128` is implied where `128` is the vPars default granularity value.

If your system (or nPartition) contains a large amount of memory (32 GB or more), you should set the granularity value to the largest amount possible to reduce the total boot time (relative to boot time in nPars/standalone) caused by the initial hardware scanning of memory. For CLM on PA-RISC platforms and for both ILM and CLM on Integrity platforms, you should choose the largest possible granularity value.

However, you should be careful when using the granularity option; using the option incorrectly can cause all the virtual partitions to not be bootable.

Further, granularity in the vPars database can only be specified during the creation of the vPars database. This means the first `vparcreate` command performed to create the database can be used to specify the granularity, but it cannot be changed after that. It cannot be changed by subsequent `vparcreate` commands nor any other commands; any change in values requires the entire vPars database to be re-created. Therefore, please read this section thoroughly.

For details on granularity values and granularity limitations, see the *vparresources* (5) manpage. The granularity section of this manpage is provided below since there are critical notes in the manpage of which you should know when planning a granularity value. These include:

- the minimum values for granularity of both ILM and CLM are 64 MBs
- the chosen granularity value(s) must be an integral power of 2 (in other words, 2^X)
- (Integrity only) there is a limit on the number of CLM granules per cell and total ILM granules you can set. Use the `vparenv` command to see the maximum possible granules for ILM and CLM for your specific system. For example:

```
# vparenv
vparenv: The next boot mode setting is "vPars".
vparenv: The ILM granule size setting is 128.
vparenv: The CLM granule size setting is 128.
vparenv: Note: Any changes in the above settings will become effective
only after the next system reboot.
vparenv: Note: The maximum possible CLM granules per cell is 64.
vparenv: Note: The maximum possible ILM granules for this system is 1024.
```

  If either of these values are exceeded when you set your granularity values, the Monitor will not boot any virtual partitions. You must rebuild your vPars database such that the number of granules related to both ILM and CLM do not exceed the numbers in your `vparenv` output.

- (PA-RISC only) Excluding the first granule, a portion of which is used by the Monitor, there must be at least one entire granule that exists below the 2 GB limit for each virtual partition. These granules below the 2 GB limit are used by kernel of each virtual partition.
- Especially for nPartitions or systems containing 32 GB or more of total memory, you should set the granule to the highest possible granule size to reduce the time in scanning the memory during the initial hardware boot.

- (Integrity only) in order for the virtual partitions in an active database to be able to boot, the granularity values in the vPars database must match those written in the system firmware.

granularity portion of vparresources (5)

```
Granularity

Memory is normally assigned to vPars in units called granules.
Exceptions are described below.  The granule values for CLM and ILM
can be different.  However, both are subject to the following rules:

      +  MOST IMPORTANT, READ CAREFULLY.  Granularity, the value of a
         granule specification, is not a resource.  Resource assignments can
         be modified, even if some resource modifications require that a
         vPar be Down.  Granularity can only be specified when creating a
         new database.  It cannot be changed thereafter.

      +  The minimum values (ILM and CLM) are 64 MB.

      +  The default values are 128 MB.

      +  The recommended specifications are described below.

      +  Any chosen granularity must be an integral power of 2, not just a
         multiple of 64.  For example, 256 is a legal value, but 192 is not.

      +  Although a granularity must be an integral power of 2, memory can
         be assigned in any multiple of that value.  For example, if the CLM
         granularity is 128 MB, it is legal to assign 384 MB of CLM to a
         vPar.

      +  Integrity systems have a platform-dependent limit to the
         number of CLM granules per cell or ILM granules that may be
         configured.  You can determine specific limits for your
         installation by using the vparenv command and examining the "The
         maximum possible xLM granules..." messages. Note: When in nPar
         mode, the vparenv command does not display the "The maximum
         possible xLM granules..." messages if the next boot mode setting is
         nPars. So in order to get these values, you have to first change
         the next boot mode setting to vPars (reboot not required) and then
         invoke the vparenv command. These values, combined with your total
         memory of each type, determine the minimum granularities you should
         specify in order to allow your vPars to boot.  For example, if you
         are allowed 1024 ILM granules and your total memory is <= 128 GB,
         you can use the default ILM granularity of 128 MB.  Or if you are
         allowed 16 CLM granules per cell, and your nPar configuration
         includes two cells each configured with 8 GB of CLM, your CLM
         granularity must be >= 512 MB.

         If the total ILM memory or CLM memory per cell exceeds that which
         can be configured in the maximum number of granules using your
         specified granularity, the vPar Monitor will not boot any vPar.  In
         this case, you must increase one or both granularities
         appropriately so that all available memory can be accommodated.
         This will require a complete reconfiguration of your database.
         Careful configuration planning will avoid this situation.


         Granularity limitations do not apply to PA-RISC platforms.
         However, there are guidelines that do apply to both PA-RISC and
         Integrity systems.  These are described next.
```

+ Recommendations for ILM and CLM granularity specifications:

  On PA-RISC platforms, each vPar needs ILM below 2 GB to load and launch its kernel. However, portions of the first granule (starting at address 0) are used for the Monitor's code and data, therefore will not be used for the kernel. **Hence, excluding the first granule, there should be at least one granule below 2 GB for each partition.** So if ILM granularity is 128 MB, the first 2 GB will consist of 16 granules. Therefore, it will be possible to load and launch the maximum supported 8 vPars. If ILM granularity is 256 MB, there are only 8 granules in the first 2 GB. The Monitor uses portions of the first one. So it will only be possible to load and launch 7 or fewer vPars. On ab Integrity server, there is no similar constraint on the maximum ILM granularity.

  For CLM on PA-RISC platforms, and for both ILM and CLM on Integrity systems, **Hewlett-Packard recommends choosing the largest possible granularity for performance reasons.** The granularity can be such that it is equal to the partition with the least amount of that memory type. For example, if the system contains 64 GB of ILM and the smallest ILM specification of any vPar is 1 GB, then ILM granularity can be 1 GB. If the system contains 64 GB of CLM per cell and the smallest CLM specification from any cell of any vPar is 4 GB, the CLM granularity can be 4 GB.

+ **For an Integrity server, the chosen granularity values must also be written to system firmware storage.** When the Monitor is started and a vPar database is loaded, the values in the database must match those in firmware, or the Monitor will not allow the database to be used.

  While in nPars mode, you should use the vparstatus and vparenv commands to verify that the database and firmware granularities are identical. If not, you must either create a new database with the correct granularities using the vparcreate -g command, or change the firmware granularities with the vparenv -g command.

Although memory is normally allocated in integral granules, some memory ranges are withheld for use by the Monitor or by firmware. The vparstatus -m command displays these ranges. Other ranges of address space are simply non-existent. Because of this fragmentation, the Monitor may assign your vPar slightly more or less than an integral granule of memory when the vPar boots.

Memory: Allocation Concepts and Notes

- The unit for the specified size of memory for the vPars commands is megabytes; parmodify uses gigabytes.
- The default memory assigned to a virtual partition is 0 MB, so you need to specify enough memory for your applications and the operating system.
- Memory is allocated in multiples of 128 MB by default. Memory assignments are also rounded up (to the next highest granule boundary).

CPU

CPU migration refers to adding CPUs to and deleting CPUs from a virtual partition. **Dynamic CPU migration** refers to migrating CPUs while the target virtual partition is running. vPars allows the assignment of most CPUs while the virtual partitions are running.

For **vPars A.04** and later, the two types of CPUs are **Boot Processor and dynamic CPUs**. This discussion begins at “CPU: Boot Processor and Dynamic CPU Definitions” on page 180. The bound and unbound CPU types in vPars A.03 and earlier no longer apply.

For additional information on using iCAP (formerly known as iCOD), including temporary-iCAP processors with vPars, see “CPU: Using iCAP (Instant Capacity on Demand) with vPars (vPars A.04 and iCAP B.07)” on page 188.

NOTE Using vPars A.03.xx and earlier syntax on a vPars A.04.xx system

Although not recommended under most circumstances, you can still use the vPars A.03.xx CPU syntax on vPars A.04.xx systems. However, the concepts and rules of boot processors and dynamic CPUs in A.04.xx will apply because the concepts and rules of bound and unbound CPUs in A.03.xx no longer apply.

For more information, see “CPU: Syntax, Rules, and Notes” on page 186.

CPU: Boot Processor and Dynamic CPU Definitions

Beginning with vPars A.04.01, the restrictions of bound CPUs have been removed as well as the terms bound and unbound. Now, there are two types of processors: boot processors and dynamic CPUs.

The **Boot Processor** is the CPU on which the OS kernel of the virtual partition was booted. There is one boot processor per virtual partition. On booting of a virtual partition, the vPars Monitor determines which CPU becomes the boot processor. Note that the specific CPU chosen as the boot processor may differ across virtual partition reboots.

Dynamic CPUs are all the other CPUs. because all CPUs, except the boot processors of each virtual partition, can be dynamically migrated. You can find which CPU is the boot processor by using the `vparstatus` command; see “Commands: Displaying Monitor and Resource Information (vparstatus)” on page 114.

Note that you can only add CPUs that are available. If you are using iCAP (formerly known as iCOD), the CPUs must be active and authorized by iCAP before you can add it to a virtual partition.

In A.04, all CPUs can process IO interrupts. See “Managing IO Interrupts” on page 187.

CPU: Specifying Min and Max Limits

The syntax to specify *min* and *max* CPUs assigned to a virtual partition remains the same.

```
-[a|m] cpu:::[min][:max]
```

where

a	is adding (used with vparcreate)
m	is modifying (used with vparmodify)
min	is the minimum number of CPUs for the virtual partition to boot and the minimum number of CPUs that must remained assigned to the partition
max	is the maximum number of CPUs that can be assigned to the virtual partition

NOTE The virtual partition must be in the down state to set the min or max value.

Total is the total count of CPUs in the virtual partition.

The total count must always be greater than or equal to *min* and less than or equal to *max*.

Please see “CPU: Syntax, Rules, and Notes” on page 186 regarding the new concepts and count rules for *min*.

Examples

- To set the minimum number of CPUs to 2:

```
keira1# vparmodify -p keira2 -m cpu:::2
```

- To set the minimum number of CPUs to 2 and the maximum to 4:

```
keira1# vparmodify -p keira2 -m cpu:::2:4
```

CPU: Adding and Deleting by Total

The basic syntax for adding and deleting CPUs is

`-a|d|m cpu::`*`num`*

where

`a|d|m`	specifies adding, deleting, or modifying the *`total`* count of CPUs
`num`	specifies the number of CPUs

NOTE The virtual partition can be either up or down when using the `cpu::num` syntax.

CPUs that are added using the `cpu::num` syntax can be deleted only by using either the

- `cpu::`*`num`* syntax or
- `cpu:`*`hw_path`* syntax (deletion by hardware path)

Total increase or decreases by num when using the -a or -d options and is set to num when using the -m option.

vparcreate

With the `vparcreate` command, you would use the -`a` option and specify as *`num`* the number of CPUs to allocate to the virtual partition.

If no cpu syntax is given on the `vparcreate` command line, the default value for total is 1.

Example

- To create the virtual partition `keira2` with 3 CPUs, set *`num`* to 3:

```
keira1# vparcreate -p keira2 -a cpu::3 ...
```

vparmodify

With the `vparmodify` command, you can either use the

- -a option to add *`num`* CPUs to the virtual partition
- -d option to delete *`num`* CPUs from the virtual partition
- -m option to modify (set) to *`num`* the number of CPUs assigned to the partition . The vPars Monitor automatically will add or delete CPUs to or from the virtual partition to reach *`num`* CPUs

Examples

- If an existing partition has 2 CPUs and you would like to set the number of CPUs to 3, you can modify the number of CPUs assigned to the partition by using the -m option and setting *`num`* to 3:

```
keira1# vparmodify -p keira2 -m cpu::3
```

  To set the number of CPUs back to two, use the -m option and set *`num`* to 2

```
keira1# vparmodify -p keira2 -m cpu::2
```

- If you would like to add 1 CPU to an existing partition, regardless of its current CPU count, you can add 1 CPU by using the -a option and setting *`num`* to 1

```
keira1# vparmodify -p keira2 -a cpu::1
```

To remove the added CPU from the partition, use the -d option and set *num* to 1

```
keira1# vparmodify -p keira2 -d cpu::1
```

CPU: Adding or Deleting by CLP (Cell Local Processor)

Similar to CLM (cell local memory), CLP (cell local processor) refers to CPUs on a specific cell. The syntax to specify CLP is

```
-[a|d] cell:cell_ID:cpu::num
```

where

`a`	is adding
`d`	is deleting
`cell_ID`	is the cell ID
`num`	is the number of CPUs from the cell to be added to or deleted from the virtual partition. Note that the *`num`* CPUs need to be available on the cell as well as the system before they can be added. To see whether they are available or already allocated, use the `vparstatus` command.

NOTE The virtual partition can be either up or down when using the CLP syntax (*`cell_ID`*`:cpu::`*`num`*).

CPUs that are added using the CLP syntax can be deleted only by using either the

- CLP syntax or
- `cpu:`*`hw_path`* syntax (deletion by hardware path)

Adding or deleting CPUs by CLP changes total accordingly, regardless of whether the virtual partition is up or down.

Examples

- To create a virtual partition using 2 CPUs from cell 6:

```
keira1# vparcreate -p keira2 -a cell:6:cpu::2 ...
```

- To increase the number of CPUs by 2 using the CPUs of cell 6

```
keira1# vparmodify -p keira2 -a cell:6:cpu::2
```

To decrease the number of CPUs by 2 using the CPUs of cell 6

```
keira1# vparmodify -p keira2 -d cell:6:cpu::2
```

CPU: Adding or Deleting by Hardware Path

The syntax for specifying by hardware path is

`-[a|d] cpu:`*`hw_path`*

where

`a`	is adding
`d`	is deleting
`hw_path`	is the *`hw_path`* (you can find the hardware path using ioscan or vparstatus -v)

NOTE The target virtual partition can be up or down when specifying by hardware path.

CPUs that are added using the hardware path syntax can be deleted only by using the

- hardware path syntax

Adding or deleting CPUs by hardware path changes total only when the virtual partition is up (unless the operation is an addition and the specified CPU is already assigned to the partition). If the virtual partition is down, total is not changed and becomes a high boundary. For example, if keira2 is down and total is set to 2, you cannot have more than 2 CPUs added by hardware path. This is for A.03-backwards compatibility.

Example

- To add the CPUs at `0/10` and `0/11` to keira2:

```
keira1# vparmodify -p keira2 -a cpu:0/10 -a cpu:0/11
```

Using both the Hardware Path Specification and CLP specification

Although you can add and delete CPUs by hardware path, to avoid confusion it is recommended that you specify by cell rather than by hardware path. The exception is for dual-CPU sockets.

For dual-CPU sockets, if you wish to have both CPUs of a socket assigned to the same virtual partition, you should specify by hardware path instead of by cell. Specifying by cell cannot guarantee that both CPUs in the socket are the CPUs chosen by the Monitor and assigned to the target virtual partition., although the Monitor will attempt to do this whenever possible.

For more information on dual-core CPU usage, see “CPU: Dual-Core” on page 190.

CPU: Syntax, Rules, and Notes

vparstatus

- When a virtual partition is down, vparstatus does not show any processor assigned as the boot processor. The boot processor is not assigned until the virtual partition is actually booted.
- If a virtual partition is down and assigned only one CPU, a CPU will be reserved by the vPars Monitor, making it unavailable. The specific CPU reserved is not determined until boot time. As a result, while the virtual partition is down, vparstatus -A, which shows available resources, will show all the possible paths of unassigned CPUs but the count of available CPUs will be one less. The count reflects the actual number of available CPUs because one CPU is reserved for the down virtual partition.

Counts Summary

- At all times, the rule of min<=total<=max is enforced.
- When adding by CLP, the total count changes whether the partition is up or down.
- When adding by hardware path, the total count changes only when the partition is up (unless the specified CPU is already assigned to the partition).
- When adding by hardware path and the partition is down, you cannot have the number of CPUs added by hardware path exceed the current total value.

Deleting CPUs Summary

- You can delete any CPU, except the current boot processor, by specifying its hardware path. Otherwise, a CPU can only be deleted using the same syntax that was used to add it to the virtual partition.
- The current boot processor can not be deleted from a virtual partition. (Use vparstatus -v to determine the current boot processor). You will need to shutdown the virtual partition and delete the desired CPU.

Managing IO Interrupts

This section described information you need if you are managing IO interrupts on a vPars-enabled system. Note that migrating interrupts should only be done by advance administrators for performance tuning.

`intctl` command

The `intctl` command is a HP-UX tool that allows you to manage IO interrupts among active processors.

For HP-UX 11i v2 and later, the software for `intctl` is part of the Core OS.

For more information, see the Interrupt Migration Product Note available at http://docs.hp.com or the *intctl* (1M) manpage.

Notes

- At boot time of a virtual partition, interrupts are processed by all the CPUs in the virtual partition.
- After boot, CPUs that are added to the virtual partition are not assigned to process IO interrupts. However, you can migrate IO interrupts to any added CPU using `intctl`.
- After boot, CPUs that are deleted from a virtual partition no longer process IO interrupts for the partition. When a CPU is deleted from a virtual partition, if the deleted CPU has IO interrupts, the IO interrupts are automatically and transparently reassigned to other active CPUs in the partition.

NOTE Repeatedly adding and deleting CPUs without a reboot of the target virtual partition may cause an imbalance in the interrupt processing load across the CPUs of the virtual partition. However, you can use `intctl` if the imbalance is not desired.

CPU: Using iCAP (Instant Capacity on Demand) with vPars (vPars A.04 and iCAP B.07)

iCAP CPUs are unlicensed CPUs. The unlicensed CPUs may be shown as available CPUs in the `vparstatus -A` output. To use iCAP CPUs, you must first purchase them; then, you can activate and assign them to a virtual partition.

For detailed information on using iCAP in vPars environment, including using earlier versions of iCAP (formerly known as iCOD) with earlier versions of vPars, see the corresponding *HP Instant Capacity User's Guide*.

Purchasing Licenses for iCAP CPUs.

To purchase licenses for any iCAP components, including iCAP CPUs, you must follow the normal iCAP process as shown in the *HP Instant Capacity User's Guide*. If you attempt to assign an iCAP CPU before purchasing the license, you will get an iCAP authorization error.

Activating and Deactivating CPUs

When you are in standalone (PA-RISC) or nPars (Integrity) mode, you can activate CPUs using the `icod_modify -a` command. (Then, while you are in the vPars environment or vPars mode, you can use `vparmodify -a` as long as you do not go above the number of Intended Active CPUs (see the Intended Active section below)).

When you are in the vPars environment or vPars mode, you can activate a CPU using `icod_modify -a`. However, this automatically activates and assigns the CPU to the local partition (the virtual partition from which the `icod_modify -a` was invoked). For example, after you have purchased 3 licenses, you can activate and assign the 3 CPUs to the local virtual partition using iCAP commands:

```
winona1# icod_modify -a 3                    /* assigns 3 CPUs to winona1 */
```

At this point, the 3 CPUs have already been added; you do not need to run `vparmodify -a cpu::3`. If you do run `vparmodify -a cpu::3`, this will add 3 more CPUs to the virtual partition (in addition to the 3 CPUs that were added with the `icod_modify` command).

Note that if you deactivate CPUs while in the vPars environment or in vPars mode using icod_modify -d, this will un-assign those CPUS from the local virtual partition.

Assigning and Unassigning CPUs

While in the vPars environment, as long as the number of CPUs assigned to your virtual partitions is less than or equal to the number of Intended Active CPUs, you can use `vparmodify` to add CPUs to your virtual partitions. As long as the number of CPUs assigned to your virtual partitions does not go below your specified vPars minimums `(cpu:::[`*min*`])`, you can delete CPUs from your virtual partitions, regardless of the number of Intended Active.

Note that as stated above, while in the vPars environment or vPars mode, using `icod_modify -a` assigns as well as activates those CPUs to the local virtual partition.

Intended Active Boundary

Using the iCAP software, the Intended Active number represents the number of licensed CPUs that could be activated within an nPartition. To view the current Intended Active number, you can use the iCAP command `icod_stat`. To change the Intended Active number, you can use the iCAP command `icod_modify`.

While in the vPars environment (PA) or vPars mode(Integrity), the total number of CPUs assigned to the virtual partitions *cannot exceed* Intended Active. This is true regardless of whether the virtual partitions are up or down. If you encounter this situation, you may need to increase Intended Active using `icod_modify -a` to activate and assign CPUs to your nPartition.

While in standalone (PA) or nPars (Integrity) mode, when the total number of CPUs assigned to the virtual partitions *exceeds* the current Intended Active number for the nPartition, iCAP allows this in the vPars database while in standalone or nPars mode but displays a warning that the virtual partitions in the vPars database will not boot. If you attempt to boot the vPars Monitor using this vPars database without increasing Intended Active using the iCAP commands, the iCAP software will disallow this and shut down any virtual partitions attempting to boot. You must reboot to standalone (PA) or nPars mode (Integrity), fix the situation so that the total number of assigned CPUs is less than or equal to the Intended Active number, and then reboot the Monitor.

When assigning to an alternate and inactive vPars database, vPars and iCAP will allow the assignments, but as in the above situation, if you attempt to boot this vPars database without increasing the Intended Active number using the iCAP commands, the iCAP software will not allow this and will shut down any virtual partitions attempting to boot. You must reboot to standalone (PA-RISC) or nPars mode (Integrity), fix the situation so that the total number of assigned CPUs is less than or equal to the Intended Active number, and then reboot the Monitor.

CPU: Dual-Core

With the PA-8800s and other dual-cores, there are two CPUs per socket. (On a cell board with four sockets, this allows 8 CPUs per cell board.) The CPUs that share the socket are called sibling CPUs.

Splitting sibling CPUs across virtual partition refers to assigning one sibling CPU to one partition and assigning the other sibling CPU to a different virtual partition. No noticeable performance degradation has been seen when splitting sibling CPUs. Due to items such as the larger L2 cache size, there actually can be a small performance boost if the siblings are split such that one of the virtual partitions has no workload. If you require consistently predictable performance, configure the virtual partitions consistently; in other words, whether you split siblings or keep them together, always do the same thing.

Determining if the system has Dual-Core CPUs

You can see the sibling and virtual partition assignment using `vparstatus -d`. If you do not have a dual-core system, the output will show dashes (-) for the sibling and assignment information:

```
# vparstatus -d
CPU                      Cell Config                    Sibling Information
path  CPU HPA            ID   Status Assigned to        Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc078000    0 E      vpuma02            -     -
0.11  0xfffffffffc07a000    0 E      vpuma01            -     -
0.12  0xfffffffffc07c000    0 E      vpuma04            -     -
0.13  0xfffffffffc07e000    0 E      -                  -     -
...
```

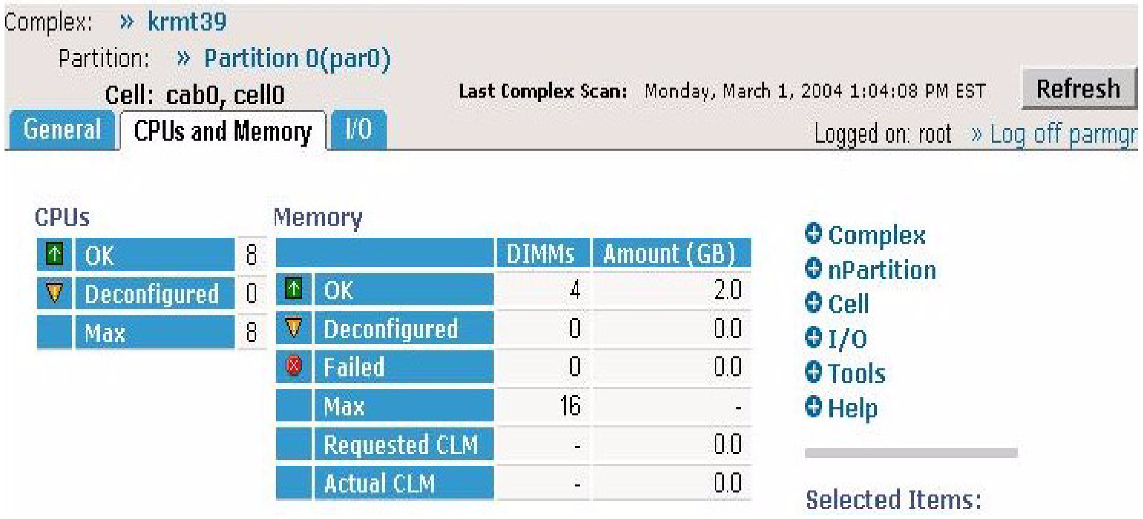
When you do have dual-core system, the `vparstatus -d` output will look similar to the following:

```
# vparstatus -d
CPU                      Cell Config                    Sibling Information
path  CPU HPA            ID   Status Assigned to        Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc070000    0 E      vpkeira1           0.11  vpkeira3
0.11  0xfffffffffc071000    0 E      vpkeira3           0.10  vpkeira1
0.12  0xfffffffffc074000    0 E      -                  0.13  vpkeira4
0.13  0xfffffffffc075000    0 E      vpkeira4           0.12  -
0.14  0xfffffffffc078000    0 E      -                  0.15  -
0.15  0xfffffffffc079000    0 E      -                  0.14  -
...
```

You can also use the `parstatus` command or `parmgr` to determine if you are running dual-cores. If the maximum number of CPUs per cell is 8, then you are running dual-cores:

```
# parstatus -c 0
[Cell]
                        CPU     Memory                              Use
                        OK/     ( GB)                         Core   On
Hardware   Actual       Deconf/ OK/                           Cell    Next  Par
Location   Usage        Max     Deconf    Connected To        Capable Boot  Num
========== ============ ======= ========= =================== ======= ====  ===
cab0,cell0 active core  8/0/8     2.0/ 0.0 cab0,bay0,chassis0  yes     yes   0
```

Figure 6-7 using parmgr to determine dual-cores


Complex: » krmt39
Partition: » Partition 0(par0)
Cell: cab0, cell0
Last Complex Scan: Monday, March 1, 2004 1:04:08 PM EST
Logged on: root » Log off parmgr

CPUs

OK	8
Deconfigured	0
Max	8

Memory

	DIMMs	Amount (GB)
OK	4	2.0
Deconfigured	0	0.0
Failed	0	0.0
Max	16	-
Requested CLM	-	0.0
Actual CLM	-	0.0

Selected Items:

Determining Sibling CPUs

Once you have determined that you have a dual-core system, the siblings have adjacent hardware paths. For example, if the `ioscan` output shows:

```
0/10    processor   Processor
0/11    processor   Processor
0/12    processor   Processor
0/13    processor   Processor
0/14    processor   Processor
0/15    processor   Processor
0/16    processor   Processor
0/17    processor   Processor
```

The hardware paths for the sibling pairs are

```
0/10 and 0/11
0/12 and 0/13
0/14 and 0/15
0/16 and 0/17
```

After vPars is installed, you can also use the vPars Monitor's `scan` command to show hardware paths.

```
MON> scan

0           CELL        sv_model=172  HPA=0xfffffffffc000000   VPAR=ALL
0/0         BUSCONV     sv_model= 12  HPA=0xffffff8020000000   VPAR=ALL
0/0/0       BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010000000   VPAR=vpar1
0/0/1       BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010002000   VPAR=vpar1
0/0/2       BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010004000   VPAR=NONE
0/0/4       BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010008000   VPAR=NONE
0/0/6       BUS_BRIDGE  sv_model= 10  HPA=0xffffff801000c000   VPAR=NONE
0/0/8       BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010010000   VPAR=vpar4
0/0/10      BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010014000   VPAR=vpar2
0/0/12      BUS_BRIDGE  sv_model= 10  HPA=0xffffff8010018000   VPAR=NONE
0/0/14      BUS_BRIDGE  sv_model= 10  HPA=0xffffff801001c000   VPAR=vpar3
0/5         MEMORY      sv_model=  9  HPA=0xfffffffffc016000   VPAR=ALL
```

```
0/10        NPROC        sv_model=  4  HPA=0xfffffffffc070000   VPAR=vpar2
0/11        NPROC        sv_model=  4  HPA=0xfffffffffc071000   VPAR=vpar3
0/14        NPROC        sv_model=  4  HPA=0xfffffffffc078000   VPAR=vpar4
0/15        NPROC        sv_model=  4  HPA=0xfffffffffc079000   VPAR=SHARED
1           CELL         sv_model=172  HPA=0xfffffffffc080000   VPAR=ALL
1/0         BUSCONV      sv_model= 12  HPA=0xffffff8120000000   VPAR=ALL
1/0/0       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110000000   VPAR=vpar1
1/0/1       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110002000   VPAR=vpar1
1/0/2       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110004000   VPAR=vpar4
1/0/4       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110008000   VPAR=NONE
1/0/6       BUS_BRIDGE   sv_model= 10  HPA=0xffffff811000c000   VPAR=vpar2
1/0/8       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110010000   VPAR=NONE
1/0/10      BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110014000   VPAR=vpar3
1/0/12      BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110018000   VPAR=vpar1
1/0/14      BUS_BRIDGE   sv_model= 10  HPA=0xffffff811001c000   VPAR=NONE
1/5         MEMORY       sv_model=  9  HPA=0xfffffffffc096000   VPAR=ALL
1/6         IPMI         sv_model=192  HPA=0xfffffffc300c0000   VPAR=ALL
1/10        NPROC        sv_model=  4  HPA=0xfffffffffc0f0000   VPAR=vpar1
1/11        NPROC        sv_model=  4  HPA=0xfffffffffc0f1000   VPAR=SHARED
1/14        NPROC        sv_model=  4  HPA=0xfffffffffc0f8000   VPAR=SHARED
1/15        NPROC        sv_model=  4  HPA=0xfffffffffc0f9000   VPAR=SHARED
```

where the following CPU pairs are siblings:

- `CPU 1/10` (owned by vpar1) and `CPU 1/11` (unassigned)
- `CPU 0/10` (owned by vpar2) and `CPU 0/11` (owned by vpar3)
- `CPU 0/14` (owned by vpar4) and `CPU 0/1`5 (unassigned)
- `CPU 1/14` (unassigned) and `CPU 1/15` (unassigned)

CPU: CPU Monitor (formerly known as LPMC Monitor)

The CPU Monitor (a part of the diagnostic tool Event Monitor Services (EMS) and not a part of the vPars Monitor) is designed to Monitor cache parity errors within the processors on the system. With its Dynamic Processor Resilience (DPR), If the CPU Monitor detects a pre-determined number of errors, the CPU Monitor will deactivate a processor for the current boot session. If the problems are severe enough, the CPU Monitor will deconfigure the socket for the next boot of the system.

Deactivation of a processor means that the OS will attempt to no longer use the processor by migrating all threads off the processor. Deactivation of a processor is *not* persistent across an OS or system reboot; CPU Monitor will deactivate the processor if it continues to detect problems.

Deconfiguration of a socket means that the EMS issues a firmware call, marking the socket for deconfiguration on the next system boot. On the next system boot, the processors on the target socket are not visible to either the OS in standalone mode or the OS instances of the virtual partitions. The deconfiguration is persistent across system boots.

Note here two items:

- a deactivation of a processor does not mean a deconfiguration of its socket. The CPU Monitor is able to determine whether the processor needs to be deactivated or whether it needs to take further action and deconfigures the socket.
- reboot of a virtual partition is not the same as a reboot of the system (the entire box or nPartition).

The exceptions to the deactivation of processors is the boot processor of each OS instance (the boot processor has a logical instance of zero; otherwise, the OS would crash) and the last CPU in a cell or nPartition. The exception to the deconfiguration of sockets is that the last remaining socket will not be deconfigured (otherwise, the system could not boot).

If any spare iCAP (formerly known as iCOD) or PPU processors are available, the necessary number of processors will be activated to replace the processors deactivated.

NOTE (A.04) On a vPars system, when a virtual partition goes down and contains deconfigured or deactivated CPUs, the Monitor will try to decommission the CPU from use and replace it with another good CPU if possible. If this is not possible, the Monitor will not allow the partition to boot until the deconfigured or deactivated CPU can be taken out of use. Following are some cases where the Monitor may not allow the virtual partition to boot:

- There is a deconfigured or deactivated CPU which has been reserved for the partition as part of the total (`cpu::`*`num`*) request and Monitor does not have any free CPUs with which to replace it. To correct this, you can delete CPUs from other partitions or from this partition.
- There is a deconfigured or deactivated CPU which has been bound to the partition due to specifying the hardware path (`cpu:`*`hw_path`*) of which the Monitor is not able to replace with another available CPU. To correct this, you can remove the CPU specified by hardware path using `-d cpu:hw_path` to allow the deconfigured or deactivated CPU to be decommissioned and replaced with another (working) CPU.
- There is a deconfigured CPU which has been reserved for the partition as part of a CLP request (cell:*`cell_ID`*:cpu:*`num`*) and there are no free CLP in that cell. To correct this, you can make available CPUs from that cell by deleting the CPUs that are part of this cell from other partitions or delete the CPUs from the cell in this partition.

The PA-8800s are dual-core sockets: they have two CPUs per socket. Deactivation happens on a processor level, but deconfiguration happens at the socket level. If a socket is deconfigured, both processors sharing the socket will be unavailable.

(Integrity only) If a CPU is **marked for deconfiguration** using an EFI command and the nPartition is not rebooted (for example, the vPars Monitor is immediately booted), the vPars Monitor will not know or indicate (including with vparstatus) that the CPU has been marked for deconfiguration and will use the CPU like any other working CPU.

Managing IO and Memory Resources with only One Virtual Partition

Adding and deleting IO and memory resources to and from a virtual partition requires the virtual partition to be in the down state. However, if you are running only one virtual partition, you will not have an OS instance from which to run the `vparmodify` commands to add or delete the IO and memory resources.

In this situation, you should do the following:

1. boot into standalone (PA-RISC) or nPars (Integrity) mode
2. add or delete the resources using `vparmodify`
3. reboot the nPartition into the vPars (PA-RISC) environment or `vPars` (Integrity) mode

.

7 CPU, Memory, and IO Resources (A.03.xx)

Managing Hardware Resources

- IO Allocation (Adding or Deleting IO Resources)
- Memory Allocation (Adding or Deleting Memory Resources)
- CPU Allocation (Adding or Deleting CPU Resources)
- CPU Monitor (deallocation and deconfiguration)

NOTE Some examples in this chapter may use a non-nPartitionable system where there is no cell in the hardware path. If using an nPartitionable system you must include the cell in the hardware path; please read "Planning, Installing, and Using vPars with an nPartitionable Server" on page 48.

IO: Concepts

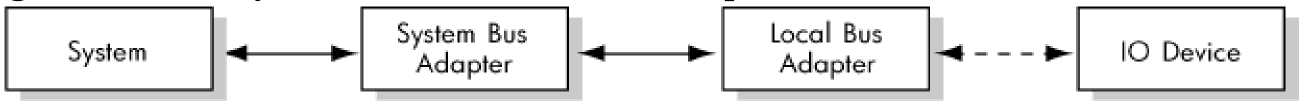
Acronyms

LBA	Local Bus Adapter
SBA	System Bus Adapter

System, Cells, SBA, LBA, Devices and Relationships

On a server, an IO device communicates to the system through the LBA and SBA. The path looks like

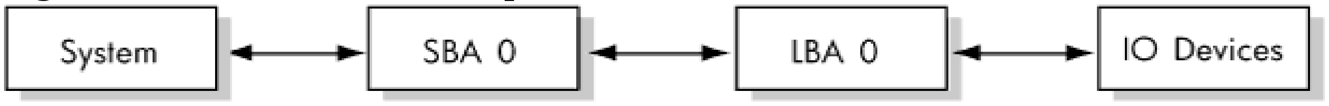
Figure 7-1 System to IO Device Relationship



This corresponds to the `ioscan` hardware path output for an IO device of `sba/lba/ ... /device`.

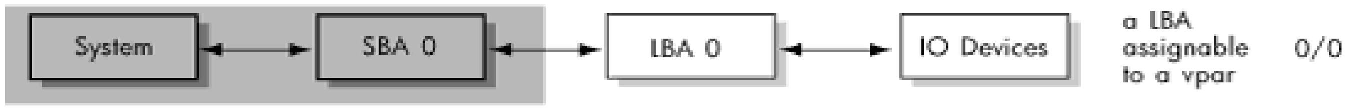
A LBA actually owns all the devices attached to it. In the example below, all the IO devices attached to LBA 0 are owned by LBA 0, and the hardware paths of those IO devices begin with 0/0 (sba/lba). (Cells are discussed later and would change the hardware path to cell_ID/sba/lba.)

Figure 7-2 LBA owns Multiple IO Devices



It is at the LBA level where vPars assigns IO. In the example below, this means that LBA 0 can be assigned to at most one virtual partition. If LBA 0 is assigned to vparN, it is implied that all IO devices attached to LBA 0 are assigned to vparN.

Figure 7-3 vPars allocates IO at the LBA Level

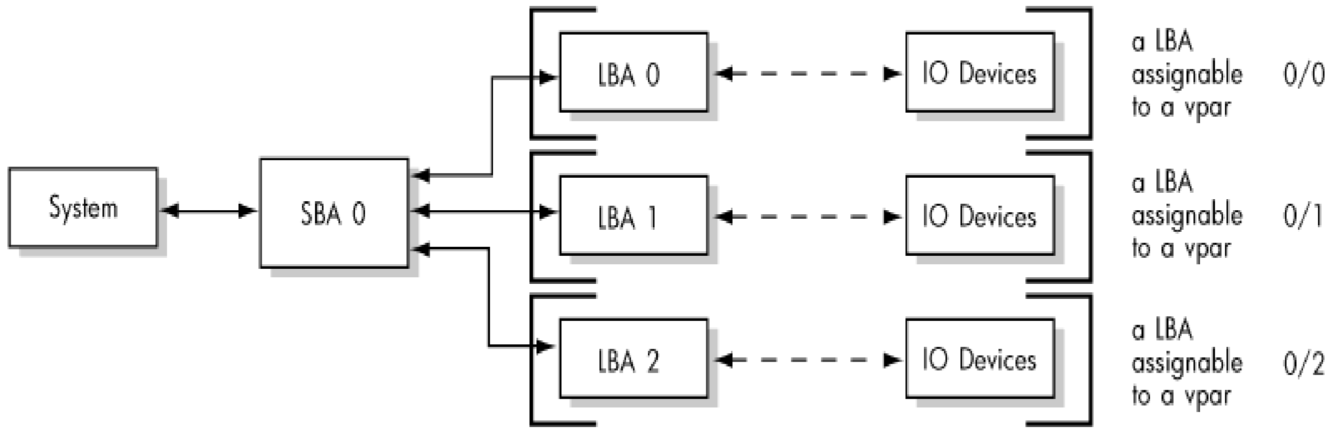


A SBA has multiple LBAs attached to it; it is a hierarchical relationship. Nevertheless, assignments in vPars remain at the LBA level, and each LBA can be assigned to a different virtual partition.

NOTE Regarding syntax and how vPars commands interpret what is specified on the command line, see “IO: Allocation Notes” on page 202. Even if there are shortcuts in assigning LBAs, vPars assigns per LBA.

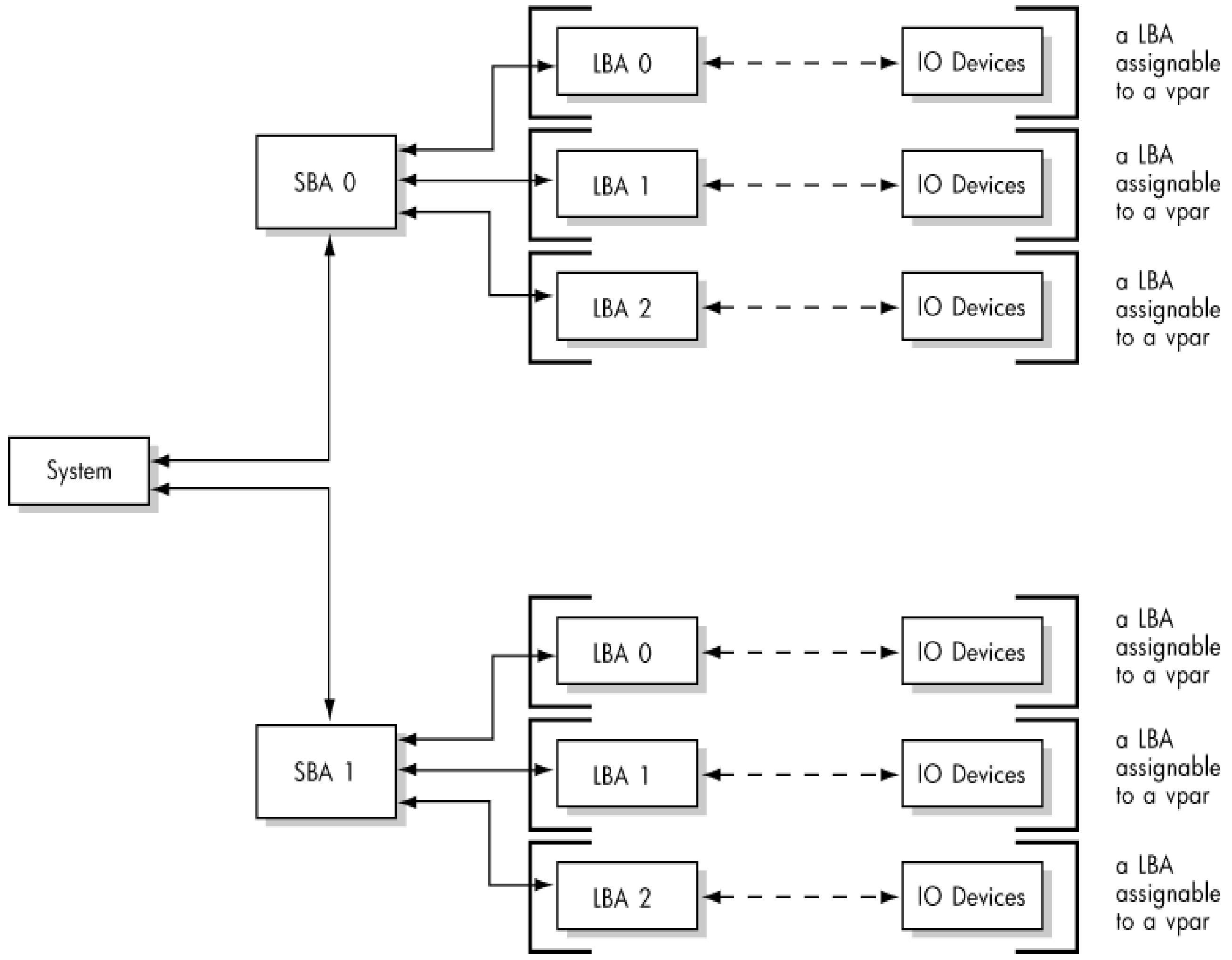
In the example below, each LBA (shown in brackets) can be assigned to a different virtual partition.

Figure 7-4 vPars allocates at LBA level not SBA level


System
SBA 0
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
0/0
a LBA assignable to a vpar
0/1
a LBA assignable to a vpar
0/2

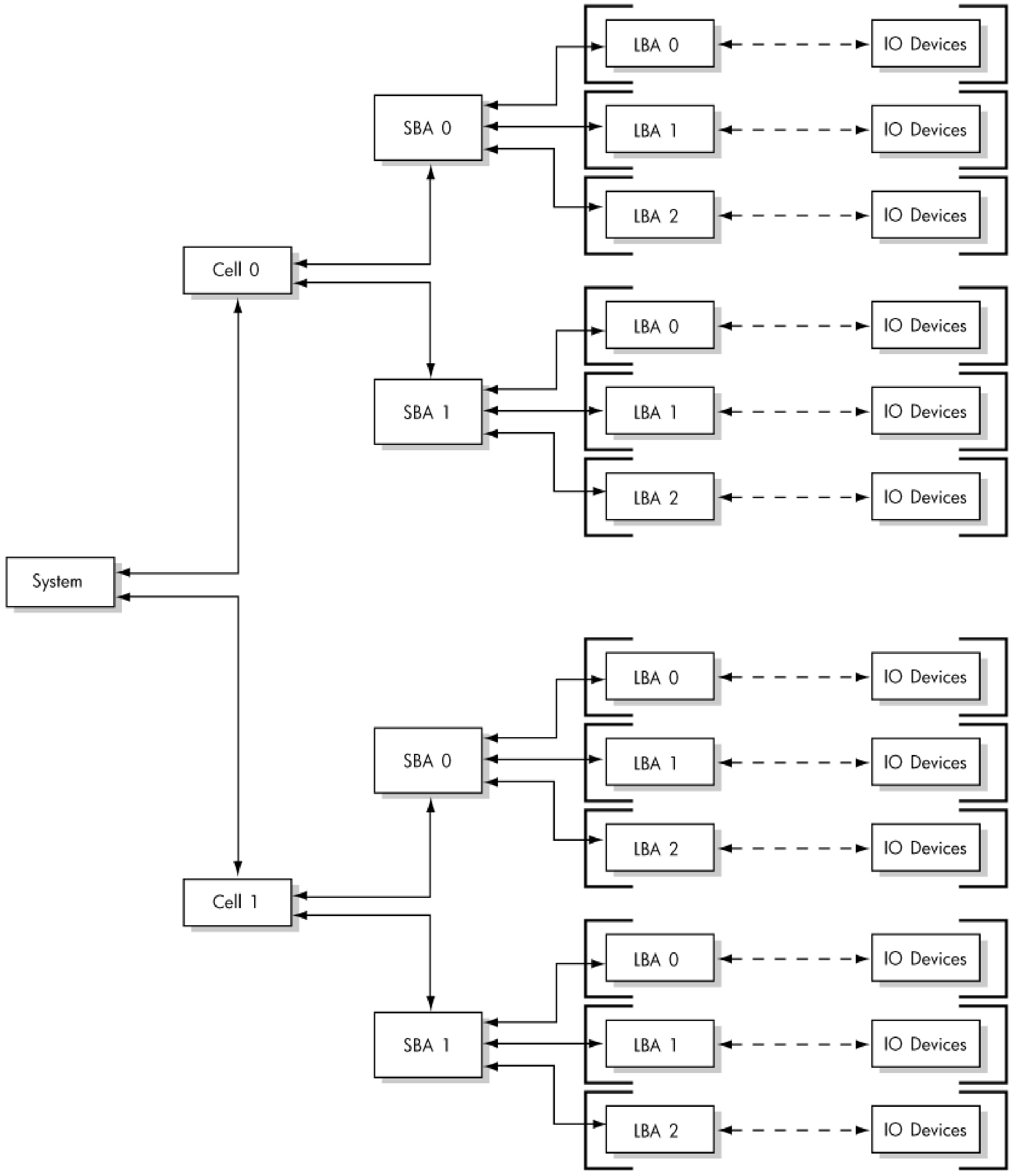
A system has multiple SBAs, but assignments remain at the LBA levels.

Figure 7-5 vPars allocates at LBA level not SBA level


SBA 0
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
a LBA assignable to a vpar
a LBA assignable to a vpar
System
SBA 1
LBA 0
LBA 1
LBA 2
IO Devices
IO Devices
IO Devices
a LBA assignable to a vpar
a LBA assignable to a vpar
a LBA assignable to a vpar

With the addition of cells (an nPartitionable server), there are more SBAs, but IO assignments remain at the LBA level:

Figure 7-6 vPars allocates at LBA level not at cell level



IO: Adding or Deleting LBAs

IO Syntax in Brief

the basic core syntax for adding or deleting IO resources is

`-a|d io:`*`hardware_path`*

where

`a`	is adding
`d`	is deleting
`hardware_path`	is the hardware path of the IO

Examples

- To add all hardware using the SBA/LBA hardware path of `1/2` to an existing partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -a io:1.2
  ```

- To remove all hardware with SBA/LBA `1/2` from partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -d io:1.2
  ```

NOTE The virtual partition must be in the `down` state to add or delete IO resources.

IO: Allocation Notes

When planning or performing IO allocation, note the following:

- **An LBA can be assigned to at most one virtual partition at any given time.**

  When you are planning your IO to virtual partition assignments, note that only one virtual partition may own any hardware at or below the LBA (Local Bus Adapter) level. In other words, **hardware at or below the LBA level must be in the same virtual partition**.

  Example

  Looking at the `ioscan` output of a rp7400/N4000, the two internal disk slots use the same LBA:

```
0/0       ba              Local PCI Bus Adapter (782)
0/0/2/0       ext_bus     SCSI C875 Ultra Wide Single-Ended
0/0/2/1       ext_bus     SCSI C875 Ultra Wide Single-Ended
```

  Therefore, you *cannot* assign one of the internal disks to partition `vpar1` and the other internal disk to partition `vpar2`; these disks must reside in the same partition.

- **Syntax Notes**

  Using vPars A.03.01 or earlier, **LBAs must be explicitly specified (included in the hardware path)**. Specifying only the SBA is not supported. If specifying only an SBA, the commands will *not* assume that all LBAs under the SBA are to be assigned; the system may actually panic.

  Beginning with **vPars A.03.02, you can specify only the SBA**. The vPars commands will assume the change applies to all LBAs under the specified SBA.

  The exception are boot disks; **boot disks are specified using the full hardware path**.

NOTE When assigning IO, if you specify a path below the LBA level (for example, `cell/sba/lba/.../device`, vPars automatically assign the LBA to the virtual partition. For example, if you specify `-a io:0/0/0/2/0.6.0` where `0/0/0` is the cell/sba/lba, the lba of 0/0/0 is assigned to the virtual partition. Further, this LBA assignment implies that all devices using `0/0/0` are assigned to the virtual partition.

The assignment rules of LBAs remain applicable: the LBA can only be owned by one virtual partition. For example, once the LBA at 0/0/0 is assigned to one virtual partition, it cannot be simultaneously assigned to any other virtual partition. Thus, if the device at `0/0/0/2/0.6.0` is assigned to a virtual partition, the LBA at 0/0/0 is assigned to that virtual partition, so the device at `0/0/0/2/0.6.0` cannot be assigned to a different virtual partition.

LBA Example

The `vparcreate` command on a non-nPartitionable system looks like:

```
#vparcreate -p vpar1 -a cpu::1 -a cpu:::1 -a mem::1024 -a io:0.0 -a io:0.0.2.0.6.0:BOOT
```

where the IO assignment is specified using the LBA level (`-a io:0.0`) and the boot disk is specified using the full hardware path (`-a io:0.0.2.0.6.0`).

For an nPartitionable system, the `vparcreate` command would look like:

```
#vparcreate -p vpar1 -a cpu::1 -a cpu:::1 -a mem::1024 -a io:0.0.0 \
-a io:0.0.0.2.0.6.0:BOOT
```

where the IO assignment is specified using the LBA level (`-a io:0.0.0.`) and the boot disk is specified using the full hardware path (`-a io:0.0.0.2.0.6.0`).

For information on using the LBA level on nPartitionable systems, also see "Planning, Installing, and Using vPars with an nPartitionable Server" on page 48.

- **SBA/LBA versus cell/SBA/LBA**

  When viewing hardware paths, note the following:

  1. The explicit specification of an LBA on a non-nPartitionable system consists of two fields: `sba/lba`
  2. The explicit specification of an LBA on an nPartitionable system consists of three fields: `cell/sba/lba`
  3. With A.03.02 and later and A.04.xx, all LBAs under a SBA are implied when explicitly specifying a SBA without specifying any LBA. Therefore, the path specified on a command line can have different meanings depending upon the vPars version, the type of server, and the user intent. For example, the path of `x/y` can mean *any* of the following:

     — sba=x, lba=y on a non-nPartitionable server running vPars A.03.01 or earlier

     — sba=x, lba=y on a non-nPartitionable server running vPars A.03.02 or later or A.04.xx

     — cell=x, sba=y on an nPartitionable server running vPars A.03.02 or later or A.04.xx

- **Supported IO**

  Check your hardware manual to verify that your mass storage unit can be used as a bootable device. If a mass storage unit cannot be used as a boot disk on a non-vPars server, it cannot be used as a boot disk on a vPars server. vPars does not add any additional capability to the hardware.

  For information on supported IO interface cards and configurations, see the document *HP-UX Virtual Partitions Ordering and Configuration Guide.*

Memory: Concepts and Functionality

Acronyms

ILM	Interleaved Memory vPars A.03.xx and A.02.xx use and assign only ILM; vPars A.04.xx allows use of ILM and CLM.

Assignments

You assign memory to a virtual partition:

- by size

  this uses the nPartition's ILM.

Within the available nPartition's ILM, you can also

- specify an address range to use

  This does not increase the amount of memory assigned to the virtual partition. The address range is a specific subset of the existing ILM amount assigned to the virtual partition. Therefore, the total amount of memory specified by ILM addresses cannot exceed the amount of ILM assigned to the virtual partition.

NOTE The virtual partition must be in the `down` state to add or delete memory resources.

Memory: Assigning By Size (ILM)

Assigning memory by specifying only size uses ILM memory. ILM memory is the only type of memory used in vPars A.03.xx and earlier. vPars A.04.xx and later can use either ILM and CLM memory.

Syntax

The basic syntax for adding or decreasing ILM resources assigned to a virtual partition is

`-[a|d] mem::`*`size`*

where

a	is adding
d	is decreasing
size	is the quantity of ILM in MBs

Examples

- To create the virtual partition `winona2` with 1024 MB of ILM:

  ```
  winona1# vparcreate -p winona2 -a mem::1024
  ```

- To add 1024 MB of ILM to an existing partition `winona2`:

  ```
  winona1# vparmodify -p winona2 -a mem::1024
  ```

- To decrease the amount of ILM assigned to partition `winona2` by 1024 MB:

  ```
  winona1# vparmodify -p winona2 -d mem::1024
  ```

Memory: Specifying Address Range

Within the already allocated memory sizes, you can specify the memory address ranges using the `mem:::`*`base`*`:`*`range`* syntax. However, this is not recommended unless you are familiar with using memory addresses and for PA-RISC systems, you should also be familiar with the 2 GB memory requirement for the HP-UX kernel and know the number of virtual partitions you will create.

For usage information, see the *vparmodify* (1M) manpage. You should select your `base:range` after consulting `vparstatus -A` to determine which ranges are available.

NOTE Specifying an address range does not increase the amount of memory assigned to the partition. Rather, it only specifies addresses to use for the already allocated memory sizes.

Therefore, all specified ranges cannot exceed the total allocated memory for the virtual partition. In other words, the sum of the ILM-specified ranges cannot exceed the total amount of ILM memory reserved for the virtual partition.

CAUTION Normally, ranges are granule-aligned (in other words, the starting address and the ending address of the range is a multiple of a granule). However, due to memory fragmentation, some of the ranges may not be granule-aligned. It is not recommended to assign such nongranule-aligned ranges as user-specified ranges (`mem:::`*`base`*`:`*`range`*). Further, future vPars releases may not support specifying ranges that are not aligned to a granule and may return an error when such ranges are assigned to a virtual partition.

2 GB Restriction (PA-RISC only)

When ranges are specified for the entire memory owned by a partition, you should ensure that at least one of the ranges is below 2 GB and is large enough to accommodate the kernel for that partition. However, other partitions also require memory below 2 GB for their kernels. Hence, you also should ensure that the specified range below 2 GB is not so large such as to preclude memory below 2 GB for the other partitions.

In general terms, the sum of the size of the kernels must be < 2 GB. To calculate the kernel sizes, see "Calculating the Size of Kernels in Memory (PA-RISC only)" on page 271.

CAUTION Not allowing enough memory for the other partitions will cause the other partitions to not boot. You can boot the partition by freeing up enough memory for the partition to boot, such as by shutting down an active partition.

If no memory ranges are below 2 GBs for a given partition, the partition will not boot.

If you use the defaults of the dynamic tunables, you will not run into the 2 GB limit. However, if you have adjusted the dynamic tunables, it is possible to run beyond the 2 GB boundary. For more information on adjusting the kernel size with dynamic tunables, see the white paper *Dynamically Tunable Kernel Parameters* at http://docs.hp.com.

Memory: Allocation Concepts and Notes

- The unit for the specified size of memory for the vPars commands is megabytes; parmodify uses gigabytes.
- The default memory assigned to a virtual partition is 0 MB, so you need to specify enough memory for your applications and the operating system.
- (vPars A.03.xx and earlier) Memory is allocated in multiples of 64 MB. Any specified size that is not a multiple of 64 MB is rounded up to the nearest 64 MB boundary. For example, if you specify 1 MB, 64 MB will be allocated.

CPU

CPU migration refers to adding CPUs to and deleting CPUs from a virtual partition. **Dynamic CPU migration** refers to migrating CPUs while the target virtual partition is running. vPars allows the assignment of most CPUs while the virtual partitions are running.

For **vPars A.03** and earlier, the two types of CPUs are **bound and unbound (floater) CPUs**. This discussion begins at “CPU: Bound and Unbound” on page 210.

NOTE Using vPars A.03.xx and earlier syntax on a vPars A.04.xx system

Although not recommended under most circumstances, you can still use the vPars A.03.xx CPU syntax on vPars A.04.xx systems. However, the concepts and rules of boot processors and dynamic CPUs in A.04.xx will apply because the concepts and rules of bound and unbound CPUs in A.03.xx no longer apply.

CPU: Specifying Min and Max Limits

The syntax to specify *min* and *max* CPUs assigned to a virtual partition remains the same.

```
-[a|m] cpu:::[min][:max]
```

where

a	is adding (used with vparcreate)
m	is modifying (used with vparmodify)
min	is the minimum number of CPUs for the virtual partition to boot and the minimum number of CPUs that must remained assigned to the partition
max	is the maximum number of CPUs that can be assigned to the virtual partition

NOTE The virtual partition must be in the `down` state to set the min or max value.

Total is the total count of CPUs in the virtual partition.

The total count must always be greater than or equal to *min* and less than or equal to *max*.

Examples

- To set the minimum number of CPUs to 2:

```
keira1# vparmodify -p keira2 -m cpu:::2
```

- To set the minimum number of CPUs to 2 and the maximum to 4:

```
keira1# vparmodify -p keira2 -m cpu:::2:4
```

CPU: Bound and Unbound

Definitions

With vPars, there are two types of CPUs: **bound** and **unbound**.

A **bound CPU** is a CPU that is assigned to and handles IO interrupts for a virtual partition. Every virtual partition must have at least one bound CPU to handle its IO interrupts.

CPUs that are not assigned to any virtual partition or that are assigned to a virtual partition but do not handle its IO interrupts are **unbound CPUs**. Unbound CPUs are sometimes called floater CPUs.

All CPUs begin as not being assigned to any virtual partition, so all CPUs begin as unbound CPUs. Using the vPars commands, you can assign CPUs to virtual partitions as bound or unbound.

You can migrate both bound and unbound CPUs, but because HP-UX cannot dynamically migrate IO interrupts, you can *dynamically* migrate only unbound CPUs.

CPU: Determining Whether to Use Bound or Unbound

When the applications within the target virtual partitions are IO intensive, use bound CPUs because only bound CPUs can process IO interrupts; specifically, with IO intensive applications there should be more bound CPUs than unbound CPUs.

If your applications are CPU intensive (and not IO intensive), use unbound CPUs so that you can easily adjust the number of CPUs via dynamic CPU migration as the demand on the virtual partition changes. Unbound CPUs provide greater flexibility of movement between virtual partitions because they can be added and removed without needing to bring down the affected partitions.

CPU: Determining When to Specify a Hardware Path for a Bound CPU

By default, the vPars Monitor chooses the hardware path of a bound CPU. However, if you need to use a specific processor, you can specify its hardware path in the vPars commands.

Generally, you do not need to specify a hardware path. The main purpose of specifying hardware paths is when you need to consider NUMA (Non-Uniform Memory Access) factors, where the distance between a CPU and memory is critical to performance.

Adding and Removing Bound CPUs

CPU Allocation Syntax In Brief

To understand how to assign CPUs, you need to understand the command syntax. Below is a brief explanation of the CPU allocation syntax for the `vparcreate` command. For complete information, see the *vparcreate* (1M), *vparmodify* (1M), and *vparresources* (5) manpages.

Syntax for `vparcreate`

The core `vparcreate` syntax for CPU allocation includes:

```
vparcreate -p partition_name [-a cpu::total] [-a cpu:::[min] [:[max] ]]] [[-a cpu:hw_path]
...]
```

where

`min` is the number of CPUs bound to *`partition_name`*. The default is 1.

`total` is the total number of bound plus unbound CPUs assigned to *`partition_name`*. The default is 1.

`max` is the maximum number of bound plus unbound CPUs that potentially can be added to the partition. The default is the number of CPUs in the server.[1]

1 <= *`min`* <= *`total`* <= *`max`*

`hw_path` is the hardware path of a bound CPU. If not specified, the Monitor chooses the hardware path.

Note on `vparmodify` Syntax

The `vparmodify` command follows a similar syntax, except that `vparmodify` allows the `-m` (modify) option as well as the `-a` (add) or `-d` (delete) option.

With the `-m` option, the number used with the `-m` is an *absolute* number. For example, `-m cpu::3` represents an absolute number of three *`total`* CPUs; in this case, it sets the *`total`* number of CPUs (bound plus unbound) to three.

With the `-a` option (as well as the `-d` option), the number used with the `-a` is a *relative* number of CPUs (relative to the number of CPUs already assigned to the virtual partition). For example, `-a cpu::3` represents three CPUs relative to the number of existing CPUs; in this case, `-a cpu::3` adds 3 *additional* unbound CPUs to the number of unbound CPUs already assigned to the partition.

1. when the Monitor is running and you are not specifying an alternate database. If the Monitor is not running or you are specifying an alternate database, the max may be a different number.

Adding a CPU as a Bound CPU

All CPUs begin as not being assigned to any virtual partition, so all CPUs begin as unbound CPUs. However, you can assign CPUs as bound CPUs to the partition by specifying the *min* number in the `-a cpu:::`*min* command line option.

Examples

- To create a virtual partition `winona2` with two bound CPUs:

  ```
  winona1# vparcreate -p winona2 -a cpu::2 -a cpu:::2
  ```

  In this example, the *total* number of CPUs assigned to the partition is two (`-a cpu::2`). Of these two CPUs, two are bound because *min* is set to two (`-a cpu:::2`).

- If the partition already exists, you can use the `vparmodify` command to set the number of bound CPUs. For example, to increase the number of bound CPUs from two to three:

  ```
  winona1# vparmodify -p winona2 -m cpu::3 -m cpu:::3
  ```

Choosing the Hardware Path of a Bound CPU

By default, the vPars Monitor chooses the hardware path of a bound CPU. However, if you need to use a specific CPU, you can specify its hardware path by using the `-a cpu:`*hw_path* option.

Examples

- In the command

  ```
  winona1# vparcreate -p winona2 -a cpu::2 -a cpu:::2
  ```

  the virtual partition `winona2` has two bound CPUs. If you want the CPU at hardware path `41` to be one of the two bound CPUs, specify the hardware path `41` (`-a cpu:41`) such that the command line is:

  ```
  winona1# vparcreate -p winona2 -a cpu::2 -a cpu:::2 -a cpu:41
  ```

- If you want to specify multiple processors, use the `-a cpu:`*hw_path* option for each hardware path. For example, if you want to specify the CPU at hardware path `41` and the CPU at hardware path `45`, the command is:

  ```
  winona1# vparcreate -p winona2 -a cpu::2 -a cpu:::2 -a cpu:41 -a cpu:45
  ```

  Note that because there are two paths specified, *min* must be greater than or equal to two. Further, because there are at least two bound CPUs, *total* must be at least two.

Removing a Bound CPU

To remove a bound CPU from a virtual partition, use the `vparmodify` command to modify the *`total`* and *`min`* parameters for the virtual partition.

Example

- If the partition `winona2` has two bound CPUs and you want only one bound CPU (and you do not want to add any unbound CPUs), set the *`total`* and *`min`* numbers to one:

  ```
  winona1# vparmodify -p winona2 -m cpu:::1 -m cpu::1
  ```

  NOTE: If you set only the *`min`* number to one and leave the *`total`* number set at two, you will still have two CPUs assigned to `winona2`. One bound CPU will be removed from the partition, but *one unbound CPU will be added* to the partition in order to maintain the *`total`* of two CPUs.

  NOTE: Because one of the value requirements for CPUs is *`min`* <= *`total`* and because command line options are processed left to right, when setting both *`min`* and *`total`* to one, you need to set *`min`* to one before setting *`total`* to one. This is accomplished by specifying the `-m cpu:::`*`min`* option before the `-m cpu::`*`total`* option.

Removing a CPU with a Specified Hardware Path

If you had specified a hardware path for a bound CPU, you would delete the specified *`hw_path`* and modify the *`min`* and *`total`* numbers.

Example

- If you have two bound CPUs and want to remove the bound CPU at hardware path 41 (and do not want to add any unbound CPUs), delete the hardware path `41`, modify *`min`* to one, and modify *`total`* number to one:

  ```
  # vparmodify -p winona2 -d cpu:41 -m cpu:::1 -m cpu::1
  ```

  NOTE: If you delete only *`hw_path`* and leave *`total`* as two and leave *`min`* as two, you will still have two bound CPUs.

NOTE When executing any operations relating to bound CPUs (adding, modifying, or deleting), the target virtual partition must be `down`.

Adding, Removing, and Migrating Unbound CPUs

For vPars A.03.xx and earlier, after *min* bound CPUs are assigned to a virtual partition, the quantity (*total* - *min*) CPUs are assigned to the partition as unbound CPUs. Therefore, to migrate unbound CPUs, specify *total* such that (*total*-*min*) is the number of unbound CPUs assigned to the target partition.

Examples

- To create the partition `winona2` with two bound CPUs and one unbound CPU, set *total* to three and *min* to two (vPars A.03.xx and earlier):

  ```
  # vparcreate -p winona2 -a cpu::3 -a cpu:::2
  ```

- To add an unbound CPU to an existing partition, use the `vparmodify` command to either modify the *total* number of CPUs (`-m cpu::`*total*) or add to the *total* number of CPUs (`-a cpu::`*total*).

  For example, to add one unbound CPU to the partition `winona2`, which already has three CPUs, two of which are bound, you can either modify *total* to four:

  ```
  winona1# vparmodify -p winona2 -m cpu::4
  ```

  or add one to *total*:

  ```
  winona1# vparmodify -p winona2 -a cpu::1
  ```

- To delete one unbound CPU from the partition `winona2`, which already has four CPUs:

  ```
  winona1# vparmodify -p winona2 -m cpu::3
  ```

  or

  ```
  winona1# vparmodify -p winona2 -d cpu::1
  ```

- Because you can dynamically migrate unbound CPUs, you can migrate an unbound CPU from one partition to another while both partitions are running. For example, if the partition `winona1` has two bound CPUs and the partition `winona2` has two bound and two unbound CPUs, you can migrate an unbound CPU from `winona2` to `winona1` using the following:

  ```
  winona1# vparmodify -p winona2 -d cpu::1
  winona1# vparmodify -p winona1 -a cpu::1
  ```

NOTE Migrating unbound CPUs may not fully complete immediately after executing the `vparmodify` commands.. For CPUs that are pending (in other words, still in the process of migrating), the vparstatus summary output will show the letter `p` next to the number of unbound CPUs and the vparstatus detailed output will show the words `(migration pending)` next to the unbound CPU. For an example, see .“Commands: Displaying Monitor and Resource Information (vparstatus)” on page 114.

For more information on CPUs, see the following:

- For information on bound and unbound CPUs, see “CPU: Bound and Unbound” on page 210.

  If you do not know which CPUs are bound CPUs and which are unbound CPUs, use the `vparstatus` command. See “Commands: Displaying Monitor and Resource Information (vparstatus)” on page 114 and the *vparstatus* (1M) manpage.

- For issues with using `vparmodify`, see the *vparmodify* (1M) manpage.

  For required partition states, see the *vparresources* (5) manpage.

Managing IO Interrupts

This section described information you need if you are managing IO interrupts on a vPars-enabled system. Note that migrating interrupts should only be done by advance administrators for performance tuning.

`intctl` command

The `intctl` command is a HP-UX tool that allows you to manage IO interrupts among active processors.

For HP-UX 11i v1, `intctl` is not installed by default with HP-UX, but you can obtain the software for `intctl` from the HP-UX Software Pack for 11i v1. Software Pack is available from the Software Pack DVD included with the HP-UX 11i OE DVDs or from the Software Depot web site at http://www.hp.com/go/softwaredepot.

For more information, see the Interrupt Migration Product Note available at http://docs.hp.com or the *intctl* (1M) manpage.

Notes

- Interrupts are processed only by bound CPUs.
- Therefore, when managing IO interrupts with `intctl`, you can manage the IO interrupts only among the bound CPUs.
- Further, disabling interrupts on a bound CPU does not convert the bound CPU into an unbound CPU.

CPU: Dual-Core

With the PA-8800s and other dual-cores, there are two CPUs per socket. (On a cell board with four sockets, this allows 8 CPUs per cell board.) The CPUs that share the socket are called sibling CPUs.

Splitting sibling CPUs across virtual partition refers to assigning one sibling CPU to one partition and assigning the other sibling CPU to a different virtual partition. No noticeable performance degradation has been seen when splitting sibling CPUs. Due to items such as the larger L2 cache size, there actually can be a small performance boost if the siblings are split such that one of the virtual partitions has no workload. If you require consistently predictable performance, configure the virtual partitions consistently; in other words, whether you split siblings or keep them together, always do the same thing.

Determining if the system has Dual-Core CPUs

You can see the sibling and virtual partition assignment using `vparstatus -d`. If you do not have a dual-core system, the output will show dashes (-) for the sibling and assignment information:

```
# vparstatus -d
CPU                      Cell Config                    Sibling Information
path  CPU HPA            ID   Status Assigned to        Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc078000    0 E      vpuma02            -     -
0.11  0xfffffffffc07a000    0 E      vpuma01            -     -
0.12  0xfffffffffc07c000    0 E      vpuma04            -     -
0.13  0xfffffffffc07e000    0 E      -                  -     -
...
```

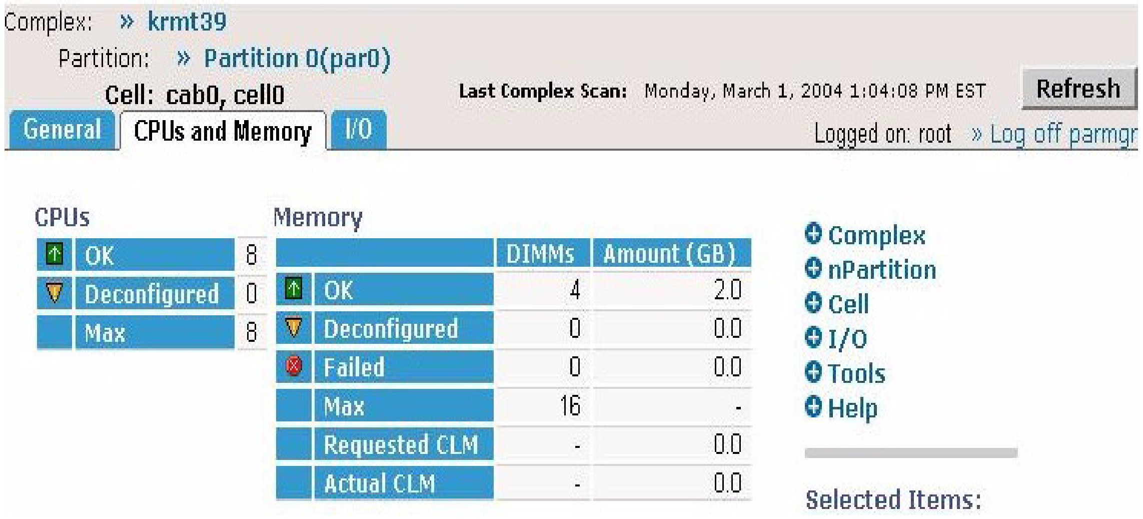
When you do have dual-core system, the `vparstatus -d` output will look similar to the following:

```
# vparstatus -d
CPU                      Cell Config                    Sibling Information
path  CPU HPA            ID   Status Assigned to        Path /vPar name
===== ================== ==== ====== ================== =======================
0.10  0xfffffffffc070000    0 E      vpkeira1           0.11  vpkeira3
0.11  0xfffffffffc071000    0 E      vpkeira3           0.10  vpkeira1
0.12  0xfffffffffc074000    0 E      -                  0.13  vpkeira4
0.13  0xfffffffffc075000    0 E      vpkeira4           0.12  -
0.14  0xfffffffffc078000    0 E      -                  0.15  -
0.15  0xfffffffffc079000    0 E      -                  0.14  -
...
```

You can also use the `parstatus` command or `parmgr` to determine if you are running dual-cores. If the maximum number of CPUs per cell is 8, then you are running dual-cores:

```
# parstatus -c 0
[Cell]
                        CPU     Memory                             Use
                        OK/     ( GB)                        Core   On
Hardware   Actual       Deconf/ OK/                          Cell   Next  Par
Location   Usage        Max     Deconf    Connected To       Capable Boot  Num
========== ============ ======= ========= ================== ======= ====  ===
cab0,cell0 active core  8/0/8     2.0/ 0.0 cab0,bay0,chassis0 yes     yes   0
```

Figure 7-7 using parmgr to determine dual-cores



Complex: » krmt39
Partition: » Partition 0(par0)
Cell: cab0, cell0
Last Complex Scan: Monday, March 1, 2004 1:04:08 PM EST
Logged on: root

CPUs	
OK	8
Deconfigured	0
Max	8

Memory	DIMMs	Amount (GB)
OK	4	2.0
Deconfigured	0	0.0
Failed	0	0.0
Max	16	-
Requested CLM	-	0.0
Actual CLM	-	0.0

Selected Items:

Determining Sibling CPUs

Once you have determined that you have a dual-core system, the siblings have adjacent hardware paths. For example, if the `ioscan` output shows:

```
0/10    processor    Processor
0/11    processor    Processor
0/12    processor    Processor
0/13    processor    Processor
0/14    processor    Processor
0/15    processor    Processor
0/16    processor    Processor
0/17    processor    Processor
```

The hardware paths for the sibling pairs are

```
0/10 and 0/11
0/12 and 0/13
0/14 and 0/15
0/16 and 0/17
```

After vPars is installed, you can also use the vPars Monitor's `scan` command to show hardware paths.

```
MON> scan

0           CELL        sv_model=172  HPA=0xfffffffffc000000   VPAR=ALL
0/0         BUSCONV     sv_model= 12  HPA=0xfffffff8020000000  VPAR=ALL
0/0/0       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010000000  VPAR=vpar1
0/0/1       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010002000  VPAR=vpar1
0/0/2       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010004000  VPAR=NONE
0/0/4       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010008000  VPAR=NONE
0/0/6       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff801000c000  VPAR=NONE
0/0/8       BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010010000  VPAR=vpar4
0/0/10      BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010014000  VPAR=vpar2
0/0/12      BUS_BRIDGE  sv_model= 10  HPA=0xfffffff8010018000  VPAR=NONE
0/0/14      BUS_BRIDGE  sv_model= 10  HPA=0xfffffff801001c000  VPAR=vpar3
0/5         MEMORY      sv_model=  9  HPA=0xfffffffffc016000   VPAR=ALL
```

```
0/10        NPROC        sv_model=  4  HPA=0xfffffffffc070000   VPAR=vpar2
0/11        NPROC        sv_model=  4  HPA=0xfffffffffc071000   VPAR=vpar3
0/14        NPROC        sv_model=  4  HPA=0xfffffffffc078000   VPAR=vpar4
0/15        NPROC        sv_model=  4  HPA=0xfffffffffc079000   VPAR=SHARED
1           CELL         sv_model=172  HPA=0xfffffffffc080000   VPAR=ALL
1/0         BUSCONV      sv_model= 12  HPA=0xffffff8120000000   VPAR=ALL
1/0/0       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110000000   VPAR=vpar1
1/0/1       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110002000   VPAR=vpar1
1/0/2       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110004000   VPAR=vpar4
1/0/4       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110008000   VPAR=NONE
1/0/6       BUS_BRIDGE   sv_model= 10  HPA=0xffffff811000c000   VPAR=vpar2
1/0/8       BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110010000   VPAR=NONE
1/0/10      BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110014000   VPAR=vpar3
1/0/12      BUS_BRIDGE   sv_model= 10  HPA=0xffffff8110018000   VPAR=vpar1
1/0/14      BUS_BRIDGE   sv_model= 10  HPA=0xffffff811001c000   VPAR=NONE
1/5         MEMORY       sv_model=  9  HPA=0xfffffffffc096000   VPAR=ALL
1/6         IPMI         sv_model=192  HPA=0xfffffffc300c0000   VPAR=ALL
1/10        NPROC        sv_model=  4  HPA=0xfffffffffc0f0000   VPAR=vpar1
1/11        NPROC        sv_model=  4  HPA=0xfffffffffc0f1000   VPAR=SHARED
1/14        NPROC        sv_model=  4  HPA=0xfffffffffc0f8000   VPAR=SHARED
1/15        NPROC        sv_model=  4  HPA=0xfffffffffc0f9000   VPAR=SHARED
```

where the following CPU pairs are siblings:

- `CPU 1/10` (owned by vpar1) and `CPU 1/11` (unassigned)
- `CPU 0/10` (owned by vpar2) and `CPU 0/11` (owned by vpar3)
- `CPU 0/14` (owned by vpar4) and `CPU 0/15` (unassigned)
- `CPU 1/14` (unassigned) and `CPU 1/15` (unassigned)

CPU: CPU Monitor (formerly known as LPMC Monitor)

The CPU Monitor (a part of the diagnostic tool Event Monitor Services (EMS) and not a part of the vPars Monitor) is designed to Monitor cache parity errors within the processors on the system. With its Dynamic Processor Resilience (DPR), If the CPU Monitor detects a pre-determined number of errors, the CPU Monitor will deactivate a processor for the current boot session. If the problems are severe enough, the CPU Monitor will deconfigure the socket for the next boot of the system.

Deactivation of a processor means that the OS will attempt to no longer use the processor by migrating all threads off the processor. Deactivation of a processor is *not* persistent across an OS or system reboot; CPU Monitor will deactivate the processor if it continues to detect problems.

Deconfiguration of a socket means that the EMS issues a firmware call, marking the socket for deconfiguration on the next system boot. On the next system boot, the processors on the target socket are not visible to either the OS in standalone mode or the OS instances of the virtual partitions. The deconfiguration is persistent across system boots.

Note here two items:

- a deactivation of a processor does not mean a deconfiguration of its socket. The CPU Monitor is able to determine whether the processor needs to be deactivated or whether it needs to take further action and deconfigures the socket.
- reboot of a virtual partition is not the same as a reboot of the system (the entire box or nPartition).

The exceptions to the deactivation of processors is the boot processor of each OS instance (the boot processor has a logical instance of zero; otherwise, the OS would crash) and the last CPU in a cell or nPartition. The exception to the deconfiguration of sockets is that the last remaining socket will not be deconfigured (otherwise, the system could not boot).

If any spare iCAP (formerly known as iCOD) or PPU processors are available, the necessary number of processors will be activated to replace the processors deactivated.

NOTE On a vPars system, for bound CPUs, the virtual partition boots with the CPU marked for deconfiguration. For unbound CPUs, the Monitor will attempt to replaced the marked CPUs with a working CPU; however, if no working CPUs are available, the Monitor automatically reduces the unbound CPU number for that virtual partition in the vPars database and allows the virtual partition to boot with the working CPUs.

The PA-8800s are dual-core sockets: they have two CPUs per socket. Deactivation happens on a processor level, but deconfiguration happens at the socket level. If a socket is deconfigured, both processors sharing the socket will be unavailable.

Managing IO and Memory Resources with only One Virtual Partition

Adding and deleting IO and memory resources to and from a virtual partition requires the virtual partition to be in the down state. However, if you are running only one virtual partition, you will not have an OS instance from which to run the `vparmodify` commands to add or delete the IO and memory resources.

In this situation, you should do the following:

1. boot into standalone (PA-RISC) or nPars (Integrity) mode
2. add or delete the resources using `vparmodify`
3. reboot the nPartition into the vPars (PA-RISC) environment or `vPars` (Integrity) mode

.

8 Version Comparisons

This section contains tables that show the differences in different version of vPars.

Feature Differences

This tables shows the differences in features among vPars A.03.xx and vPars A.04.xx on both PA-RISC and Integrity platforms.

Table 8-1 Feature Differences

vPars Version	A.03.xx	A.04.xx on PA-RISC	A.04.xx on Integrity
Product Number	T1335AC	T1335BC	
Obtaining vPars Bits	Order at HP Software Depot (http://www.hp.com/go/softwaredepot)		
Media Format	CD	DVD	
Obtaining vPars Documents	Download from HP Documentation Site (http://docs.hp.com)		
Platforms	PA-RISC		Integrity
Server Boot Firmware	ISL		EFI
Server Mode Specifications	N/A		`vPars` or `nPars`
Server Mode Specification Possible Methods	N/A		• Shell> fsN: fsN:\> vparconfig reboot *mode* • MON> reboot *mode* • HP-UX# vparenv -m *mode*
Boot vPars Monitor Steps	ISL> hpux /stand/vpmon		Shell> fs0: fs0:\>vparconfig reboot vPars ... Shell> fs0: fs0:\> hpux vpmon
Booting to standalone mode special steps	none		set to nPars mode before booting nPartition
PRI and ALT saved across nPars-mode/standalone and vPars-mode/vPars changes	yes		no
Console Port Assigned to First Partition Requirement	yes		no
Console vcn listed in ioscan output	yes		no
Monitor Commands	all available		not all available
New vPars commands since A.03	no		EFI level: vparconfig HP-UX level: vparenv, vparefiutil
SCSI parms commands	`vparutil`	`mptutil`	
Supported IO and LAN cards	as listed in vPars Ordering and Configuration Guide (OC)		same as supported on Integrity nPartition servers; exceptions noted in OC

Table 8-1 Feature Differences (Continued)

vPars Version	A.03.xx	A.04.xx on PA-RISC	A.04.xx on Integrity
vparboot -I	vparboot -p *target_vpar* -I *ignite_ux_server,WINSTALL_path*		vparboot -p *target_vpar* -I
LAN card used in vparboot -I	source vpar		target vpar
Update w/in vPars environment from A.03	no, must do outside of vPars environment as standalone	yes, must use ignite-ux depot and update-ux	no, only PA-RISC->PA-RISC is allowed
IO Syntax	Same		
Memory Syntax	by size by address range	by size by address range by cell	
Memory Types	ILM	ILM or CLM	
vPars Database Granularity Values	not allowed	allowed	
CPU Types	Bound or Unbound	Boot Processor or Dynamic	
CPUs Able to Process Interrupts	Only Bound CPUs	At boot time, all CPUs Otherwise, any (using intctl)	
Recommended CPU Specifications and Assignments	• min specification • by total • by hardware path • max specification NOTES: total=bound + unbound number of CPUs assigned by hw_path is subset of min count	• min specification • by total • by hardware path • max specification • by cell local processor NOTES: total=boot processor + assigned by Monitor + by cell local processor + by hardware path	
OS	11i v1	11i v2	
vPars GUI	yes	not available	
Parmgr	Required for Install	Not Required for Install	

Table 8-1 Feature Differences (Continued)

vPars Version	A.03.xx	A.04.xx on PA-RISC	A.04.xx on Integrity
PPU Supported Products	Percent Utilization	Percent Utilization Active CPU	

Transitioning from vPars A.03 to vPars A.04 (CPU Syntax and Rules)

The values in a vPars database that were created using vPars A.03.xx and ported to vPars A.04.xx will have the following A.04.xx meanings for those values; likewise, using vPars A.03.xx syntax on a vPars A.04.xx system has the following A.04.xx meanings.

This table also summarizes the A.04.xx syntax and rules

Table 8-2 CPU Syntax from A.03 to A.04

vPars A.03 Syntax	=>	vPars A.04 Syntax
cpu:::*min* • number of bound CPUs • minimum number of CPUs assigned to the partition. • cannot change *min* while virtual partition is up		cpu:::*min* • minimum number of CPUs assigned to the partition • cannot change *min* while virtual partition is up
cpu:*hw_path* • specify CPU by hardware path • number of CPUs assigned by hardware path is subset of *min*		cpu:*hw_path* • specify CPU by hardware path • when a virtual partition is down, the number of CPUs assigned by hardware path must be less than or equal to the pre-existing *total* setting
cpu::*total* • total = bound + unbound CPUs		cpu::*total* • total = (number of CPUs assigned by hw_path) + (number of CPUs assigned by cell) + (number of CPUs assigned by vPars Monitor) NOTE: when the virtual partition is booted, one of the CPUs from the operand list above becomes the boot processor. NOTE: from a user's point of view, the same CPU could be considered under more than one category. However, the vPars Monitor will categorize and count a CPU only once. Use vparstatus -v to see the CPU details
max as in cpu:::*min*:*max* • *max* is maximum number of CPUs that can be assigned to the virtual partition		*max* as in cpu:::*min*:*max* • same meaning as in A.03

Table 8-3 CPU Rules from A.03 -> A.04

vPars A.03 Rules	=>	vPars A.04 Rules
dynamic migration • only unbound CPUs can be migrated while a virtual partition is up		dynamic migration • all CPUs except the boot processor can be migrated while a virtual partition is up
count rules • number of cpu_assigned by hw_path <= *min* • *min*<= *total*<= *max*		count rules • *min* <= *total* • total = (number of CPUs assigned by hw_path) + (number of CPUs assigned by cell) + (number of CPUs assigned by vPars Monitor) NOTE: when the virtual partition is booted, one of the CPUs from the operand list above becomes the boot processor. NOTE: from a user's point of view, the same CPU could be considered under more than one category. However, the vPars Monitor will categorize and count a CPU only once. Use vparstatus -v to see the CPU details • *total* <= *max* • when a virtual partition is down, the number of CPUs assigned by hardware path must be less than or equal to the pre-existing *total*
adding or decreasing the *total* count of CPUs assigned to a virtual partition is performed by only • modifying *total* directly (cpu::*total*)		adding or decreasing the *total* count of CPUs is performed by any of the following: • modifying *total* directly (cpu::*total*) • adding or decreasing by hw_path (cpu:*hw_path*) — if the virtual partition is up, total will be adjusted automatically (unless the CPUs are already assigned to the partition) — if the virtual partition is down, the number of CPUs assigned by hardware path must be less than or equal to the pre-existing *total* if this is the case, you should first modify *total* to allow the additional cpu. • adding or decreasing by CLP — total will be adjusted automatically
adding by num (-a cpu::*num*) • decreasing requires decreasing by num (-d cpu::*num*)		adding by num (-a cpu::*num*) • decreasing requires decreasing by total (-d cpu::*num*) or by hw_path (-d cpu:hw_path)

Table 8-3 CPU Rules from A.03 -> A.04 (Continued)

vPars A.03 Rules	=>	vPars A.04 Rules
adding by hw_path (-a cpu:*hw_path*) • deletion requires deleting by hw_path (-d cpu:*hw_path*) • this does not adjust `total` • this does not adjust `min` • requires the partition to be `down`		adding by hw_path (-a cpu:*hw_path*) • deletion requires deleting by hw_path (-d cpu:*hw_path*) • this does adjust `total` if the virtual partition is up (unless the CPU is already assigned to the partition); otherwise, if the virtual partition is down, the number of CPUs assigned by hardware path must be less than or equal to the current `total` • partition can be up or down
deleting by hw_path (-d cpu:hw_path) • can only delete CPU added by hw_path		deleting by hw_path (-d cpu:hw_path) • can delete any CPU (except the boot processor)
adding by cell • not available in A.03		adding by cell (-a cell:*cell_ID*:cpu::*num*) • requires deleting by either of the following: — by cell (-d cell:*cell_ID*:cpu::*num*) — by hw_path (-d cpu:*hw_path*)
boot processor • not available in A.03		boot processor • cannot be deleted while the virtual partition is up Note that the actual boot processor can change across virtual partition reboots. Use the `vparstatus -v` command to determine which CPU is the current boot processor
vparstatus output when virtual partition is down • shows all bound CPUs assigned to the partition		vparstatus output when virtual partition is down • shows CPUs assigned by hardware path or cell
IO interrupts • only bound CPUs can handle IO interrupts • `intctl` (interrupt control) applies to only the bound CPUs		IO interrupts • at boot: all CPUs can handle IO interrupts • after boot: — when CPUs are removed from a virtual partition, the IO interrupts are automatically reassigned to the other active CPUs — when CPUs are added to a virtual partition, the IO interrupts are not automatically processed by the added CPUs — `intctl` can be applied to all CPUs

9 Crash Processing and Recovery

Crashing and Recovery Processes

- Crash Processing
- Network and Tape Recovery
- Expert Recovery

Crash Processing

Crash processing for a virtual partition is similar to the crash processing of a non-vPars OS: the OS is quiesced, portions of memory are written to disk, and in the case of vPars, resources are released to the Monitor.

When the Monitor crashes, a Monitor dump is created. By default, kernel dumps are not saved.

When there is a HPMC or MCA or when a TOC is issued, the virtual partitions are launched for crash processing. On PA-RISC, this occurs after saving the Monitor dump, and on Integrity, this occurs before saving the Monitor dump.

When you enter the crash user interface:

- To let the crash processing continue, do nothing.
- To enter the crash user interface, press any key on the console.

NOTE HP recommends that you let crash processing continue.

Crash User Interface

If you enter the crash user interface, you will see messages similar to the following on the console:

```
Virtual Partition Activity at Time of Crash
   partition 0 (vpar1):       active
   partition 1 (vpar2):       active
   partition 2 (vpar3):       inactive

The active partitions will be invoked to perform
crash handling.  A soft reset will then be generated
to allow additional debugging.

TO OVERRIDE THIS BEHAVIOR, PRESS A KEY WITHIN 10 SECONDS....

CRASH PROCESS STOPPED.

Crash Command Menu
1.  Examine memory contents
2.  Continue with default crash handling
3.  Cause Monitor crash dump to different device
4.  Soft reset the machine (memory preserved)
5.  Hard reset the machine (memory not preserved)
6.  Launch partition 0 (vpar1) for crash processing
7.  Launch partition 1 (vpar2) for crash processing
Enter number (1-7):
```

The menu choices mean:

1. displays memory from <`starting address`> for <*n*> 32-bit words. For example:

   ```
   Enter Address: 0x1000 4
   0x00001000 0x00000000 0x1200a000 0xaa400000 0x00000000

   Enter Address: quit
   ```

2. continues with the default crash handling

3. (PA-RISC only) allows you to chose an alternate device to which the Monitor dump is written. The alternate device must contain the pre-allocated file `/stand/vpmon.dmp`. The file `vpmon.dmp` is automatically created in `/stand` of a partition's boot disk by the vPars startup script.
4. soft resets the current hard partition[1].
5. hard resets the current hard partition.
6. boots the specified virtual partition for crash processing
7. boots the specified virtual partition for crash processing

If you chose to invoke a virtual partition for crash processing or to examine memory contents, you will be returned to this menu after those actions are completed (assuming no new crash event is encountered).

Directory Location and Filenames

On PA-RISC, the Monitor dump is written to the pre-existing file called `/stand/vpmon.dmp`.

On Integrity, the Monitor dump is written to a file called `vpmon.dmp` that is created in the EFI partition of the Monitor boot disk. The file will be at `fsN:/efi/hpux/vpmon.dmp`.

When the virtual partition that owns the Monitor boot disk is booted, the following files are created automatically in `/var/adm/crash/vpar` (where *n* is a number representing the *n*th occurrence of a dump):

`vpmon.`*n*	copy of the executable image of the Monitor at the time of the dump
`vpmon.dmp.`*n*	copy of the Monitor dump file
`summary.`*n*	an analysis of the crash including PIM info for each processor

NOTE On PA-RISC, the file `/stand/vpmon.dmp` is a special file. Do not delete, move, rename, or modify this file. If you need to look at the contents of the Monitor dump file, use the `vpmon.dmp.n` file located in `/var/adm/crash/vpar`.

Monitor Dump Analysis Tool

Because the vPars Monitor is not a HP-UX kernel, you cannot use a kernel dump analysis tool to examine a Monitor dump file. Contact your HP Support Representative to analyze the Monitor dump file.

Kernel Dumps

If a TOC (transfer of control) or HPMC (High Priority Machine Check) for the entire hard partition is generated, a kernel dump will not automatically be saved to `/var/adm/crash` for those partitions that have not previously had a kernel dump occur. You can save their dumps to `/var/adm/crash` by performing the following on each of those virtual partitions:

Step 1. (11i v2 only) For HP-UX 11i v2 and therefore vPars A.04, the savecrash processing has changed. Instead of copying the kernel file that was in use during the crash, the directory `/stand/crashconfig` is copied. Therefore, prior to executing the `crashconf` and `savecrash` steps below, create the `/stand/crashconfig` directory using

```
# kconfig -s crashconfig
```

1. A soft reset (option `4`) is not supported on a Superdome.

Or if the kernel configuration used in the last boot is different from the current kernel configuration, use the -c option. For example, if the saved kernel configuration is named kc.custom, the command is

```
# kconfig -c kc.custom crashconfig
```

For more information on using the `kconfig` command, see the manpages *kconfig* (5) and *kconfig* (1M)

Step 2. Obtain of list of dump devices, noting the `DEVICE` and `OFFSET` information:

```
# crashconf -v
DEVICE          OFFSET(kB)  SIZE(kB)   LOGICAL VOL.   NAME
--------------  ----------  ---------  -------------  --------------
31:0X022000         314208     4194304    64:0X000002  /dev/vg00/lvol2
```

The `DEVICE` is `31:0X022000`, and the `OFFSET` is 314208.

Step 3. Map the minor number from the `DEVICE` information to a device file:

```
# ls -l /dev/dsk | grep "022000"
brw-r-----   1 bin        sys         31 0x022000 Oct 13  2001 c2t2d0
```

The corresponding device file is `/dev/dsk/c2t2d0`.

Step 4. Using the `OFFSET` information and the device file, save the dump to `/var/adm/crash`:

```
# savecrash -r -f -D /dev/dsk/c2t2d0 -O 314208
```

Network and Tape Recovery

This section covers different methods of network and tape recovery on vPars systems. For information on performing a recovery using:

- `make_net_recovery` within a vPars-environment, see “Using make_net_recovery within a vPars Environment (vPars A.04, A.03 and earlier)” on page 236
- `make_tape_recovery` outside of a vPars-environment, see “Using make_tape_recovery Outside of a vPars Environment” on page 238
- `make_tape_recovery` within a vPars-environment in conjunction with an Ignite-UX boot server, see “Using make_tape_recovery and Dual-media Boot” on page 242
- `make_tape_recovery` within a vPars-environment, see “Using make_tape_recovery within a vPars-environment (vPars A.03.03)” on page 243
- golden images for recovery, see the whitepaper *Using Golden Images with Virtual Partitions*
- peripheral boot devices, see the whitepaper *Booting, Installing, Recovery, and Sharing in a vPars Environment from DVD / CDROM / TAPE / Network*
- Ignite-UX, including `make_net_recovery` and `make_tape_recovery`, see the manual *Ignite-UX Administration Guide* and the manpages *make_tape_recovery* (1m) and *make_tape_recovery* (1m).

The whitepapers listed above are available at the HP Documentation web site:

http://docs.hp.com/hpux/11i/index.html#Virtual%20Partitions

NOTE Using vparboot -I with Ignite-UX on PA-RISC systems

When using Ignite-UX version C.06.xx or later, note that the bootable kernel path has changed

from `/opt/ignite/boot/WINSTALL` in Ignite-UX B.05.xx and earlier
to `/opt/ignite/boot/Rel_B.11.NN/WINSTALL` in Ignite-UX C.06.xx and later

Thus, when using Ignite-UX C.06.xx or later, you must specify the absolute path for the bootable kernel for the `vparboot -I` command line.

For more information and an example, see “(PA-RISC only) The WINSTALL Boot Kernel Paths with Different Versions of Ignite-UX and the vparboot -I command” on page 24.

Using `make_net_recovery` within a vPars Environment (vPars A.04, A.03 and earlier)

Archiving Virtual Partition

`make_net_recovery` works the same for making archives of both non-vPars and vPars systems.

Recovering a Virtual Partition from a Running Virtual Partition

To recover a virtual partition, perform the following from a running virtual partition. (In these examples, the partition `winona1` is running and the target partition `winona2` is the partition being recovered.)

1. Record the following:

   a. the autoboot attribute of the target partition using `vparstatus`. You may need to set it back to this state in the last step.

   ```
   winona1# vparstatus -p winona2
   [Virtual Partition]
                                                               Virtual Partition Name
   State Attributes
   ========================== ===== ==========
   winona2                   Down  Dyn,Manual
   ```

   b. the contents of the AUTO file in the LIF area of the primary boot disk pointed to by stable storage. Use the `lifcp` command to see the contents.

2. Set the `TERM` environment variable to `hpterm`. For POSIX shell, the command is

   ```
   winona1# export TERM=hpterm
   ```

3. Boot the target partition and point the boot kernel to use your Ignite-UX server (assume the Ignite server's IP is ww.xx.yy.zz):

   ```
   winona1# vparboot -p winona2 -I ww.xx.yy.zz,/opt/ignite/boot/WINSTALL
   ```

4. Run the Ignite-UX recovery as you would on a hard partition not running vPars, entering the data (boot disk and LAN) of the target partition.

5. After the target partition has been recovered:

   a. if the autoboot attribute has been changed, set it back to what was recorded in the first step. For example, to set the autoboot attribute back to `manual`:

   ```
   winona1# vparmodify -p winona2 -B manual
   ```

   b. if needed, set the AUTO file back to its original contents that were recorded in the first step using the `lifrm` and `lifcp` commands.

Recovering All the Virtual Partitions of a Hard Partition

To recover all the virtual partitions within a hard partition, first recover the virtual partition whose boot disk is the disk set as the primary path within system-wide stable storage. Once the virtual partition is recovered, recover the other virtual partitions one by one. (There is no way to recover all partitions simultaneously.)

To recover the initial virtual partition:

1. From the BCH prompt, boot the hard partition using the Ignite-UX server (assume the Ignite server's IP is ww.xx.yy.zz):

```
BCH> bo lan.ww.xx.yy.zz install
interact with IPL? N
```

2. From the Ignite-UX window, select "Install HP-UX".
3. Enter the network data using the data for the virtual partition that owns the boot disk that is set as the primary path within system-wide stable storage.
4. Select `Recovery Archive Configuration -> Go`

After this virtual partition is recovered, recover the remaining partitions using the instructions in "Recovering a Virtual Partition from a Running Virtual Partition" on page 236.

Using `make_tape_recovery` Outside of a vPars Environment

The creation of `make_tape_recovery` tapes is supported on vPars-enabled servers. However with vPars A.04.01, A.03.01, and A.03.02, recoveries using these tapes must be done outside of the vPars environment; they cannot be used to recover a system from within a virtual partition. For example, the tape cannot be used with the `vparboot -I` command.

NOTE The exception to this is using a dual-media boot. For information on using a dual-media boot, see "Using make_tape_recovery and Dual-media Boot" on page 242.

The following sections describe:

- archiving and recovering a virtual partition
- archiving and recovering a virtual partition using another virtual partition as the Ignite-UX server

Assumptions

In the following example, it is assumed that the version of vPars in the tape archive is the same as that installed in the other virtual partitions.

If the virtual partition being recovered owns the system or hard partition's primary boot path, and if changes have been made to the vPars configuration since `make_tape_recovery` was run, then the recovered vPars database (`/stand/vpdb`) will be out of date.

If the recovered configuration is out of date, the recovery will require *one* of the following additional steps:

- Boot the vPars Monitor from an alternate boot disk with the current vPars database. When the recovered virtual partition is booted, the database will be synchronized with the current configuration.
- Recover an up-to-date database file from a backup before booting the vPars Monitor.
- Boot the vPars Monitor and the recovered partition, and then update the configuration with `vparmodify`, `vparcreate`, and `vparremove`.

Archiving and Recovering a Virtual Partition

Archiving the Virtual Partition(s) This section describes how to create the recovery tape.

NOTE

- To recover a single virtual partition from a tape, all active virtual partitions must be shutdown.

  The exception to this is using a dual-media boot. For information on using a dual-media boot, see "Using make_tape_recovery and Dual-media Boot" on page 242.
- The `make_tape_recovery` command is not a backup utility. The virtual partition should be backed up separately. A well thought out backup strategy should be part of every recovery plan. Your normal backups may be required to recover the virtual partition. Test your recovery plan to make sure it works properly

---

1. The virtual partition must have a tape drive attached, as it will be used in step 4 to boot the tape. The tape drive must be available to the nPartition at boot time.

   ```
   # make_tape_recovery -A -a /dev/rmt/1mn
   ```

   The following is archived to tape when `make_tape_recovery` is run:

   a. The data necessary to recover the virtual partition on a "cold" system (nothing running on it, including vPars). This includes the system filesystems (root, /stand, etc.)

   b. The files required by vPars: the vPars Monitor (the default is `/stand/vpmon`) and the vPars database (the default is `/stand/vpdb`).

2. You must document the following information about the system (not the virtual partition) and must be available in hard copy or electronically in an accessible location not on the system itself.

   a. The primary and alternate boot paths. You must get this information from the boot console handler (`BCH`). You cannot retrieve this information via the `setboot` command from a virtual partition.

   b. The contents of the `AUTO` file in the boot `LIF`. An example is `lifcp /dev/rdsk/<dev>:AUTO -` where `/dev/rdsk/<dev>` is the boot device for the system, the primary boot path in part (a). Note: If you attempt this within a virtual partition you must do it from the virtual partition that has access to the device, as only one virtual partition will be able to see it.

Recovering the Virtual Partition(s)

3. Shutdown all virtual partitions and reset the nPartition.

4. Boot the `make_tape_recovery` tape created in step (1) in the nPartition. Note that nothing is running in the nPartition. You are booting without vPars at this point.

5. Once the recovery tape has completed recovering the system, you will still be running without vPars. To re-enable vPars perform the following steps:

   a. Correct the primary and alternate boot paths if necessary by using setboot. This works at this step because vPars is not active.

   b. Correct the autoboot setting if necessary (`mkboot -a "string" /dev/rdsk/<dev>:AUTO` where `/dev/rdsk/<dev>` is the boot device for the system and "`string`" is the contents of the `AUTO` file from step (2)(b) above. The device file name may be different from that found in step (2)(a).

6. Reboot the nPartition. The vPars Monitor will start automatically if step (5) completed correctly. Any virtual partition that has been defined to autoboot will boot at this stage. You may have to manually start any virtual partitions not configured to autoboot. The vPars Monitor will only start automatically if the `AUTO` file was originally configured to do so. If not, you will boot up in standalone mode.
7. Once the virtual partition has started you can complete any other recovery of application data, or other virtual partitions.

Archiving and Recovering a Virtual Partition Using Another Virtual Partition as the Ignite-UX Server

Archiving the Virtual Partitions Using a Virtual Partition as the Ignite-UX Server The following steps describe how one or more virtual partitions can be archived using `make_tape_recovery`. These first three steps describe how to create a disaster recovery tape.

1. One of the virtual partitions is an Ignite server. Its root disk is the one that is booted first, when the vPars Monitor is booted. It has the vPars Monitor (`/stand/vpmon`) and the vPars database (`/stand/vpdb`) that is used to bring up virtual partitions in the nPartition. It must also have a tape drive which will be used by `make_tape_recovery` in step (3). This tape drive will also be used in step (4) to boot the tape created in step (3) thus it must be available to the nPartition at boot time.
2. The Ignite server makes recovery tapes of all the other virtual partitions using `make_net_recovery`. This is done when the Ignite server is running in a virtual partition, archiving the other virtual partitions while they are running.
3. The Ignite server makes a recovery tape of the system it is running on using `make_tape_recovery` and "normal" filesystem recovery tapes. This is performed while the Ignite server is running in a virtual partition. It allows the Ignite server to archive itself while the other virtual partitions are running production work. The tape created by `make_tape_recovery` in this step will have:
   a. the data necessary to recover the Ignite server on a "cold" system (nothing running on it, including vPars).
   b. the files required by vPars: the vPars Monitor (/stand/vpmon) and the vPars database (/stand/vpdb).
   c. the files created in step (2) by make_net_recovery. These files will be used to recover the other virtual partitions in step (8).
   d. normal filesystem recovery archive of the Ignite server.

Recovering the Virtual Partitions Using one of the Virtual Partitions as the Ignite-UX Server

4. The nPartition must have a tape drive available to boot from. Note that nothing is running in the nPartition. Boot the `make_tape_recovery` tape created in step (3) in an nPartition. The system is being booted without vPars at this point.
5. Recover the Ignite server that was archived to tape in step (3). This is done using the `make_tape_recovery` tape that was booted in step (4) along with normal filesystem recovery.
6. Reboot the nPartition, this time using the root disk that was recovered in step (5). Stop at the `MON>` prompt.
7. Use vparload at the `MON>` prompt to load the virtual partition recovered in step (5). This is the Ignite server.
8. Use `vparboot -I` to recover the other virtual partitions using the `make_net_recovery` files created in step (2).
9. There may be normal filesystem recoveries that need to be done to fully recover the virtual partitions after they are booted in step (8).
10. Modify the autoboot string (using `mkboot -a ...`) so that the virtual partitions will autoboot at the next system boot.
11. Reboot the nPartition to test if all the virtual partitions come up as expected.

Using `make_tape_recovery` and Dual-media Boot

A dual-media boot allows you to boot the target partition using the Ignite-UX server and then recover using the tape device.

1. If needed, make sure the target virtual partition is in the down state. For example, if it is up, shutdown the virtual partition:

   ```
   winona2# shutdown -hy 0
   ```

2. Boot the virtual partition to the Ignite-UX server. For example,

   - With Ignite-UX versions prior to C.06.xx

     ```
     winona1# vparboot -p winona2 -I ww.xx.yy.zz,/opt/ignite/boot/WINSTALL
     ```

   - With Ignite-UX versions C.06.xx and later:

     ```
     winona1# vparboot -p winona2 -I ww.xx.yy.zz,/opt/ignite/boot/Rel_B.11.11/WINSTALL
     ```

3. When the main Ignite-UX menu is displayed, select `Install HP-UX`.
4. When the `User Interface and Media Options` screen is displayed, select `Media only installation` from the `Source Location Options`, and `Advanced Installation` from `User Interface Options`, then select `OK`.
5. From the `Media Installation Selection`, select `Boot from CD/DVD, Recover from Tape`. Note that there does not need to be a CD or DVD in the server. Select `OK`.
6. From the `Tape Drive Selection` menu, select the appropriate tape drive, and press the `Enter`. Once you enter the Ignite-UX `itool` screen, proceed as normal with the recovery.

Using `make_tape_recovery` within a vPars-environment (vPars A.03.03)

Beginning with vPars A.03.03, vPars supports tape drives. This includes recovery of a virtual partition within a vPars environment and without using an Ignite-UX server as a boot helper.

Requirements and the `BOOT` Attribute

To use the tape device during a recovery, you must meet the following requirements:

- the tape device must be owned by the target virtual partition (the partition that is being recovered)
- the tape drive must be connected through an external SCSI device or be internal to the machine
- the tape device must be an explicitly specified resource with the attribute **`TAPE`**. This is similar to specifying the boot device with the attribute `BOOT`. For example:

  ```
  # vparcreate -p winona2 -a io:1/0/14/0/0/4/0.5.0:TAPE
  ```

  or

  ```
  # vparmodify -p winona2 -a io:1/0/14/0/0/4/0.5.0:TAPE
  ```

  Please note that when modifying io resources, the target virtual partition must be in the down state.

  The `vparstatus -v` command should show the tape device with the `TAPE` attribute:

  ```
  # vparstatus -p winona2 -v
  [Virtual Partition Details]
  Name:         winona2
  .
  .
  .
  [IO Details]
     1.0.14
     1.0.12
     1.0.14.0.0.4.0.10.0.0.0.0.0  BOOT
     1.0.14.0.0.4.0.5.0.0.0.0.0  TAPE
  .
  .
  .
  ```

Archiving

The process of creating an archive is the same as for non-vPars OS instances:

```
# make_tape_recovery -A -a /dev/rmt/1mn
```

Recovering

To begin recovery, boot the target virtual partition using the tape drive as specified with the TAPE attribute:

```
winona1# vparboot -p winona2 -B TAPE
```

Then proceed with the recovery as normal.

NOTE The system may appear but is actually not hung when booting from tape due to the increased time it takes to load a kernel from tape instead of from disk.

Expert Recovery

When you are performing Expert Recovery, you need to remember the following:

- You can no longer read from or write to system-wide stable storage using `setboot`. See “Boot||Shut: Setboot and System-wide Stable Storage” on page 137.
- `mkboot` modifies the LIF area, but vPars does not use the LIF area to boot a virtual partition. See “mkboot and LIF files” on page 26 and “The AUTO File on a Virtual Partition” on page 143.
- When you need to use boot options (for example, -is for single-user mode or -lm for LVM maintenance mode), please read the sections “Boot||Shut: Booting a Virtual Partition” on page 132 and “Boot||Shut: Other Boot Modes” on page 147.
- The HP-UX shell commands `shutdown` and `reboot` apply to the OS instance of a virtual partition and do not shutdown or reboot the Monitor.
- There is no way to halt the hard partition from the MON> prompt. See “Boot||Shut: Shutting Down or Rebooting a Virtual Partition” on page 133.

10 vPars Flexible Administrative Capability (vPars A.03.03 and vPars A.04.02)

This chapter discusses the concepts and tasks on using the vPars Flexible Administrative Capability feature (formerly called Primary-Admin vPars Security). With this feature, you can specify vPars administration capabilities for zero, one, or more designated virtual partitions. Only superusers within the designated virtual partitions can perform the vPars administration commands that affect other virtual partitions; a superuser within a non-designated virtual partition can perform only operations that affect itself.

Additionally, for this flexible administrative capability to work, all the virtual partitions must be running the same version of vPars.

NOTE **Applying RBAC to vPars A.04.01 Whitepaper**

You can apply the existing HP-UX Security feature RBAC (Role-based Access Control) to vPars A.04.01. For information, see the whitepaper titled *Securing Virtual Partitions with HP-UX Role-Based Access Control* available at the HP Documentation web site: http://docs.hp.com.

HP-UX Security and other Security Applications

This feature is not intended to replace existing HP-UX security or security applications. It provides as a way to limit intentional access but is not intended to substitute security or security application that eliminate malicious or unintentional circumvention of commands or provide kernel level security isolation. This feature is intended to address tighter vPars administration control requirements in certain customer deployments.

Synopsis

The vPars **Flexible Administrative Capability** feature restricts the usage of specific vPars commands such that they can be successfully executed from only designated virtual partitions.

The specific vPars commands that are restricted are those that can alter other virtual partitions, such as `vparmodify` or `vparreset`.

The designated virtual partitions are known as **designated-admin virtual partitions** and are designated by being explicitly added to the designated-admin virtual partitions list. Virtual partitions that are not in the list are considered **non-designated-admin virtual partitions**. When a superuser executes a command that affects another partition from within a non-designated-admin virtual partition, the command will fail.

When the flexible administrative capability feature is ON (enabled), a virtual partition can be added to (or deleted from) the list from either the Monitor prompt without a password or the HP-UX shell prompt by superusers who know the flexible administrative capability password.

The flexible administrative capability feature can be set to either ON (enabled) or OFF (disabled) but only from the vPars Monitor prompt (MON>).

Terms and Definitions

target partition

This is the virtual partition that is affected when a vPars command is executed. For example, in the command:

```
# vparmodify -p winona2 -a cpu::1 ...
```

an attempt is made to add a CPU to winona2, so winona2 is the target virtual partition. The argument of the -p option is the target partition.

local partition

This is the virtual partition from which a vPars command is executed. For example, in the command:

```
winona1# vparmodify -p winona2 -a cpu::1
```

assuming the HP-UX shell prompt contains the hostname, the vparmodify command is executed from winona1, so winona1 is the local virtual partition. winona2 is the target partition.

designated-admin virtual partition

This is a virtual partition that is allowed to perform vPars commands that affect other virtual partitions. For example, assume the flexible administrative capability feature is ON and the following command is executed:

```
winona1# vparmodify -p winona2 -a cpu::1
```

Because this command affects another virtual partition (winona2), the local virtual partition winona1 must be a designated-admin virtual partition in order for this command to be successful.

non-designated-admin virtual partition

This is a virtual partition that is *not* allowed to perform vPars commands that affect other virtual partitions. For example, assume the flexible administrative capability feature is ON and the following command is executed:

```
winona1# vparmodify -p winona2 -a cpu::1
vparmodify: Error: Only Designated-Admin virtual partitions can perform this
operation on winona2.
```

Because this command affects another virtual partition (winona2), if the local virtual partition winona1 is a not a designated-admin virtual partition, the command will not be successful.

designated-admin virtual partition list

This is the list of virtual partitions that are currently set as designated-admin virtual partitions. If a virtual partition is in this list or added to this list, it is a designated-admin virtual partition. If a virtual partition is not in this list or is deleted from this list, it is a non-designated-admin virtual partition. How to add or delete virtual partitions from this list is discussed in a later section.

Whenever the flexible administrative capability mode is set to ON, the designated-admin virtual partition list will be empty.

flexible administrative capability mode

When this mode is set to ON, only designated-admin virtual partitions are allowed to successfully execute vPars commands that affect other partitions. For example, in the command:

```
winona1# vparmodify -p winona2 -a cpu::1
```

this command will check whether the local virtual partition (winona1) is a designated-admin virtual partition or if the target partition is the local partition. If either condition is true, the command is executed. If not, the command is denied execution. How to set the mode is discussed in a later section.

By default, the flexible administrative capability mode is OFF.

flexible administrative capability password

From the Monitor prompt, when setting the flexible administrative capability mode to ON, you will be prompted to set a flexible administrative capability password.

From the HP-UX shell, if a superuser attempts to add or delete a virtual partition from the designated-admin virtual partition list, the superuser will be prompted for the flexible administrative capability password that was set.

When the flexible administrative capability mode is OFF, there is no flexible administrative capability password.

This designated administration feature relies on high quality passwords that are not easily guess-able, so please choose a password wisely.

Flexible Administrative Capability Commands

The flexible administrative capability commands are:

• **monadmin**	executed from the vPars Monitor (MON>)
• **vparadmin**	executed from the HP-UX shell

MON> `monadmin`

`monadmin` is a vPars Monitor command that allows you to

- display whether the vPars flexible administrative capability feature is ON (enabled) or OFF (disabled)
- set or reset the flexible administrative capability mode
- specify a virtual partition to be added or deleted to or from the designated-admin virtual partition list
- list which virtual partitions are currently in the designated-admin virtual partition list (in other words, are set as designated-admin virtual partitions)

Basic Syntax and Usage

```
monadmin [-S on|off] | [-a|-d partition_name] | [-l]
```

`-S on|off`

Sets the flexible administrative capability mode either ON (enabled) or OFF (disabled).

— ON and OFF are not case-sensitive.
— The flexible administrative capability mode can be set only at the Monitor prompt (MON>).
— By default, the mode is OFF.

When the mode is set to ON, the list of designated-admin virtual partitions is empty, regardless of any previous settings. Therefore, when the flexible administrative capability mode is set to ON, all virtual partitions are non-designated-admin virtual partitions regardless of whether it is running or not and regardless of its state during the previous time the mode was set to ON.

Because the designated-admin virtual partition list is empty when the mode is set to ON, you will need to add any virtual partitions you wish to be designated-admin virtual partitions to the designated-admin virtual partition list every time you set the mode to ON. You can add or delete virtual partitions to or from the designated-admin virtual partition list from either the Monitor prompt using monadmin (`MON> monadmin`) or the HP-UX Shell prompt using vparadmin (`# vparadmin`). `vparadmin` is discussed in the next section.

This process allows you to change the mode at the MON> prompt and maintain a deterministic setting for the virtual partition even if virtual partitions are running.

When the mode is set to ON, monadmin will prompt for a new password. (If the mode is already ON, you will receive a message stating that you already having enabled the flexible administrative capability.)

When the mode is set to OFF, the above restrictions of the vPars command execution are turned off. It may appear that all virtual partitions are considered to be designated-admin virtual partitions. Once you set the mode to OFF, the designated-admin virtual partition list is deleted as well as the flexible administrative capability password.

-a|-d *partition_name*

Adds or deletes a virtual partition to or from the designated-admin virtual partition list. No flexible administrative capability password is required here; passwords are required only at the HP-UX shell prompt.

-l

Lists all the virtual partitions that are currently in the designated-admin virtual partition list. When the mode is set initially to ON, the designated-admin virtual partition list will be empty, so all virtual partitions are non-designated-admin virtual partitions. You can use the `monadmin -a` to add virtual partitions to the list. Alternatively, at the HP-UX shell prompt, you can use `vparadmin -a` to add virtual partitions to the list.

Without any options

displays whether the vPars flexible administrative capability feature is ON (enabled) or OFF (disabled).

NOTE The Monitor prompt (MON>) is not protected. Any person who knows the GSP/MP login and password can access the console and therefore the MON> prompt. Using monadmin, the GSP/MP user can change the flexible administrative capability mode and the designated-admin virtual partition list. Console access is restricted using the GSP/MPs login and password.

# vparadmin

`vparadmin` is a vPars command that allows you to

- display whether the vPars flexible administrative capability feature is ON (enabled) or OFF (disabled)
- change the flexible administrative capability password
- specify a virtual partition to be added or deleted to or from the designated-admin virtual partition list
- list which virtual partitions are currently in the designated-admin virtual partition list (in other words, are set as designated-admin virtual partitions)

Basic Syntax and Usage

```
vparadmin [-C] | [-a|-d partition_name] | [-l]
```

-C

Changes the flexible administrative capability password. It will ask you for the existing password before allowing you to change it.

-a|-d *partition_name*

Adds or deletes a virtual partition to or from the designated-admin virtual partition list. You will need to provide the flexible administrative capability password.

`-l`

Lists all the virtual partitions that are currently in the designated-admin virtual partition list. Use `vparadmin -a` to add virtual partitions to the list. Note that `vparstatus` does *not* show any flexible administrative capability information.

Without any options

displays whether the vPars flexible administrative capability is ON (enabled) or OFF (disabled).

Persistence across Monitor Reboots

If the flexible administrative capability mode is not changed from ON to OFF across Monitor reboots and the specific conditions are met (see below), the flexible administrative capability mode will remain and the virtual partitions designated as designated-admin virtual partitions will remain as designated-admin virtual partitions. In other words, the flexible administrative capability mode and the designated-admin virtual partition list are persistent across Monitor reboots when the following conditions are met:

- the disk from which the Monitor is booted is owned by a virtual partition when flexible administrative capability changes are performed.
- the virtual partition in the above condition is up when flexible administrative capability changes are made

These conditions are required because the Monitor cannot perform a write into the vPars database; only a running virtual partition can perform the write.

Therefore, persistence across Monitor reboots will not be maintained under any of the following conditions:

- The Monitor boot disk is not owned by any of the virtual partitions or no virtual partitions are rebooted from the same disk. In this case, the vPars database of the Monitor’s boot disk will not be updated.
- All virtual partitions are down such that any flexible administrative capability changes cannot be written to the vPars database.
- The virtual partition whose boot disk is the disk from which the Monitor is booted is down when flexible administrative capability changes are made.

vPars Commands

When the flexible administrative capability mode is ON, the vPars flexible administrative capability feature restricts the vPars commands such that you can alter another virtual partition only if you execute the command from a partition that is in the designated-admin virtual partition list. When you execute the command from a non-designated-admin virtual partition, if the command alters another virtual partition, it will not be allowed. (A.03.03 only: this is true even if the vPars commands are applied to an alternate database).

The table below shows the results when the flexible administrative capability feature is ON.

Table 10-1 Flexible Administrative Capability Impact on vPars Commands

vPars command	**Executed from a ...**	
	designated-admin virtual partition	**non-designated-admin virtual partition**
`vparboot`	allowed	not allowed
`vparcreate`	allowed	not allowed
`vparremove`	allowed	not allowed
`vparmodify`	allowed	not allowed unless target partition is the local virtual partition
`vparreset`	allowed	not allowed unless target partition is the local virtual partition
`vparutil`	allowed	not allowed unless target partition is the local virtual partition

`vparenv -m`
`vparenv -g`
	allowed	not allowed

The remaining vPars commands, such as `vparstatus`, are allowed by all virtual partitions regardless of the flexible administrative capability mode because they do not alter other virtual partitions.

NOTE **When the Target Partition is the Local Partition**

If you use a command that alters a virtual partition but you execute it from the partition itself (in other words, the target partition equals the local virtual partition), this is allowed. For example,

```
winona2# vparmodify -p winona2 -a cpu::1
```

Because you are not modifying another virtual partition, this will be allowed even if winona2 is not a designated-admin virtual partition.

`vparcreate`

While flexible administrative capability is on, when you create a virtual partition, the target partition will be a non-designated-admin virtual partition. You cannot `vparcreate` a virtual partition as a designated-admin virtual partition. After the `vparcreate` command, to change the non-designated-admin virtual partition to a designated-admin virtual partition, you will need to add the partition to the designated-admin virtual partition list using the `vparadmin -a` command.

`vparstatus`

`vparstatus` does not show whether a virtual partition is in the designated-admin virtual partition list; you need to use `vparadmin -l`.

Example Monitor Scenario (`monadmin`)

Below describes examples that include (from the Monitor):

- turning on the flexible administrative capability feature (which will include setting the password)
- adding virtual partitions to the designated-admin virtual partition list

For this section, let’s assume we have the virtual partitions winona1, winona2, and winona3.

Turning On The Flexible Administrative Capability Feature

Turning on the flexible administrative capability feature for the first time is performed usually after

- at least one virtual partitions has been created (so that you have a vPars database)
- the vPars Monitor has been booted (so that you have the vPars product running)

Also, this allows you to have at least one virtual partition to be a designated-admin virtual partition, which allows you to vparcreate, vparboot, vparreset, and vparmodify other partitions if needed.

Assuming we have already installed the vPars product, created virtual partitions, and have booted the Monitor, we can set the flexible administrative capability feature to ON. When the flexible administrative capability feature is set to ON, you will also be asked for a password that will be the new flexible administrative capability password. The old password is not required.

```
MON> monadmin -S on
Enter the vPar flexible administrative password:
Re-enter to confirm:
```

Adding Virtual Partitions to the Designated-admin Virtual Partition List

At this point, no virtual partitions have been added to the designated-admin virtual partition list. Let’s add winona1 to the list.

```
MON> monadmin -a winona1
```

After we have completed the designated-admin list, let’s boot all the virtual partitions.

```
MON> vparload -all
```

Example HP-UX Shell Scenario (`vparadmin`)

Below describes examples that include (from the HP-UX shell):

- a command successfully executed
- a command not executed due to the flexible administrative capability feature
- adding a virtual partition to the designated-admin virtual partition list
- deleting a virtual partition from the designated-admin virtual partition list
- listing the virtual partitions in the designated-admin virtual partition list
- changing the flexible administrative capability password
- determining whether you are in flexible administrative capability

For this section, let's assume we have the virtual partitions winona1, winona2, and winona3.

A Command Successfully Executed

Because winona1 is in the designated-admin virtual partition list. We can execute a command from winona1 that alters winona2:

```
winona1# vparmodify -p winona2 -a cpu::1
```

Note that you will not see any flexible administrative capability messages when flexible administrative capability is allowed; you will only see flexible administrative capability messages when access is denied.

A Command Not Executed Due to the Flexible Administrative Capability Feature

However, because winona2 is not in the designated-admin virtual partition list. We cannot successfully execute a command from winona2 that alters another partition:

```
winona2# vparmodify -p winona1 -a cpu::1
vparmodify: Error: Only Designated-Admin virtual partitions can perform this operation on winona1.
winona2# vparmodify -p winona3 -a cpu::1
vparmodify: Error: Only Designated-Admin virtual partitions can perform this operation on winona3.
```

Note that if the target partition is the local virtual partition, then the partition is only altering itself and not altering another partition, so this will be allowed.

```
winona2# vparmodify -p winona2 -a cpu::1
```

Adding a Virtual Partition to the Designated-admin Virtual Partition List

If we want the non-designated-admin virtual partition winona2 to be able to alter other virtual partitions, we need to add it to the designated-admin virtual partition list. This can be done from any virtual partition, but you will need to know the flexible administrative capability password.

```
winona2# vparadmin -a winona2
password:
Virtual partition winona2 is added to the Designated-Admin virtual partitions list.
```

Deleting a Virtual Partition to the Designated-admin Virtual Partition List

If later we want to remove winona2 from the designated-admin virtual partition list, we can do this from any virtual partition:

```
winona1# vparadmin -d winona2
Password:
Virtual partition winona2 is deleted from the Designated-Admin virtual partitions list.
```

Listing the Virtual Partitions in the Designated-admin Virtual Partition List

We can verify that winona2 has been removed from the designated-admin virtual partition list. This can be performed from any partition.

```
winona1# vparadmin -l
---------- Designated-Admin virtual partitions ----------
winona1
```

Only winona1 is displayed as a designated-admin virtual partition. Since winona2 (and winona3) are not in the list, they are not designated-admin virtual partitions.

Changing the Flexible Administrative Capability Password

We can change the flexible administrative capability password with the `vparadmin -C` command. Note that there is no `-C` option in the `monadmin` command, although you will be asked for a new password when you set the mode from OFF to ON.

The `vparadmin -C` command can be executed from any virtual partition; you will need to know the old password to be able to change the password.

```
winona2# # vparadmin -C
Old password:
New password:
Re-enter new password:
The vPar flexible administrative password successfully changed.
```

Determining whether Flexible Administrative Capability is ON or OFF

When you are within a Unix shell, the easiest method to determine whether the flexible administrative capability mode is ON (enabled) is to use the `vparadmin` command.

When you are not in flexible administrative capability, you will receive the “disabled” message:

```
# vparadmin
The virtual partition flexible administrative capability is disabled.
```

When you are in flexible administrative capability, you will receive the “enabled” message:

```
# vparadmin
The virtual partition flexible administrative capability is enabled.
```

11 Virtual Partition Manager (A.03.xx)

This chapter provides an overview of the Virtual Partition Manager (`vparmgr`), which provides a GUI to the vPars commands. This chapter includes:

- About the Virtual Partition Manager
- Starting the Virtual Partition Manager
- Using the vPars Graphical User Interface (GUI)
- Stopping the Virtual Partition Manager

For more detailed information, see the Virtual Partition Manager online help.

NOTE **Discontinuance of the Virtual Partition Manager (vparmgr):**

The Virtual Partition Manager is not available for vPars A.04.01 and later.

About the Virtual Partition Manager (`vparmgr`)

The virtual partition manager (`vparmgr`) provides an easy to use graphical interface to the vPars command utilities. Using `vparmgr`, you can perform the following tasks:

- create a new virtual partition
- modify an existing virtual partition, including
  - modify attributes such as boot path and kernel path
  - add or remove processors
  - add or remove memory
  - add or remove IO local bus adapters
- delete an existing virtual partition
- display the resources assigned to each virtual partition
- display the status of each virtual partition
- display logs of vPars activity
- boot a virtual partition
- reset a virtual partition

The virtual partition manager has special features to save you time and effort:

- with a command-line parameter, you can start `vparmgr` directly in the task that you want to perform
- after using the graphical interface to select the task and options that you want to perform, vparmgr can show you the command line that would perform the operation directly.

Starting the Virtual Partition Manager

Before you can start the virtual partition manager, vPars must be installed and running. For information on installing vPars, see "Installing, Updating, or Removing vPars and Upgrading Servers with vPars" on page 61. For information on installing the virtual partition manager, see "Installing and Removing vPars-related Bundles" on page 62.

After vPars is installed and running, you must boot at least one virtual partition to a HP-UX kernel. You can then start the virtual partition manager in that virtual partition by executing the command `vparmgr`

```
/opt/vparmgr/bin/vparmgr [-h]
/opt/vparmgr/bin/vparmgr -tcreate
/opt/vparmgr/bin/vparmgr -tmodify|par_details -pvp_name
```

With no arguments, the vparmgr graphical user interface is launched. You can perform all vparmgr operations from the GUI, as discussed below under using the graphical user interface.

Options

`-h`

displays usage instructions

`-t` *`create`*

creates a new virtual partition

`-t` *`modify`*

modifies an existing virtual partition. You must specify which virtual partition to modify, using the *`vp_name`* parameter.

`-t`*`par_details`*

displays the status, attributes, and resources of a virtual partition. You must specify which virtual partition to display, using the *`vp_name`* parameter.

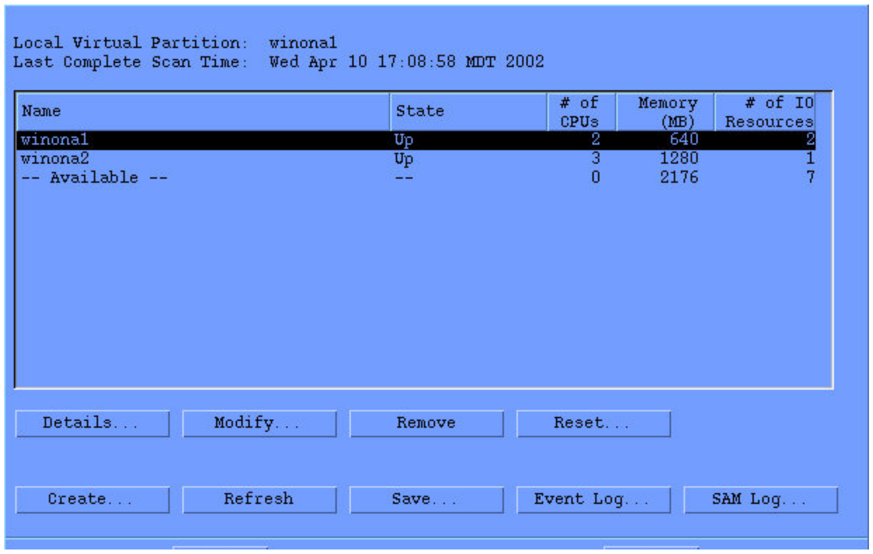
`vp_name`

the name of a virtual partition

Using the vPars Graphical User Interface (GUI)

When the vparmgr GUI starts, it displays the virtual partition status screen.

Figure 11-1 vPars GUI Status Screen



```
Local Virtual Partition:   winona1
Last Complete Scan Time:   Wed Apr 10 17:08:58 MDT 2002
```

Name	State	# of CPUs	Memory (MB)	# of IO Resources
winona1	Up	2	640	2
winona2	Up	3	1280	1
-- Available --	--	0	2176	7

This displays the status of all of the virtual partitions and available resources on the system. To perform an action on an object (a virtual partition or a set of available resources), click on the object in the list and then click the button corresponding to the action that you want to perform on that object. For more information about this screen, select the `virtual partition status page` from the navigation links on the left side of the Virtual Partition Manager online help.

Each vparmgr screen works in a similar fashion. Select the object, then click a button to act on the object. Some buttons do not require a selected object. For example, the `refresh` button will refresh the display, using the latest data available from the vPars Monitor.

The online help provides a more detailed information for each vparmgr screen. Clicking any screen's `help` button will launch a web browser to display the help specific to that screen. For more information about using the online help system, click `using help` from the navigational links on the left side of the Virtual Partition Manager online help.

Stopping the Virtual Partition Manager

To exit vparmgr, click the `Exit` button on the virtual partition status screen.

A LBA Hardware Path -> Physical IO Slot Correspondence (PA-RISC only)

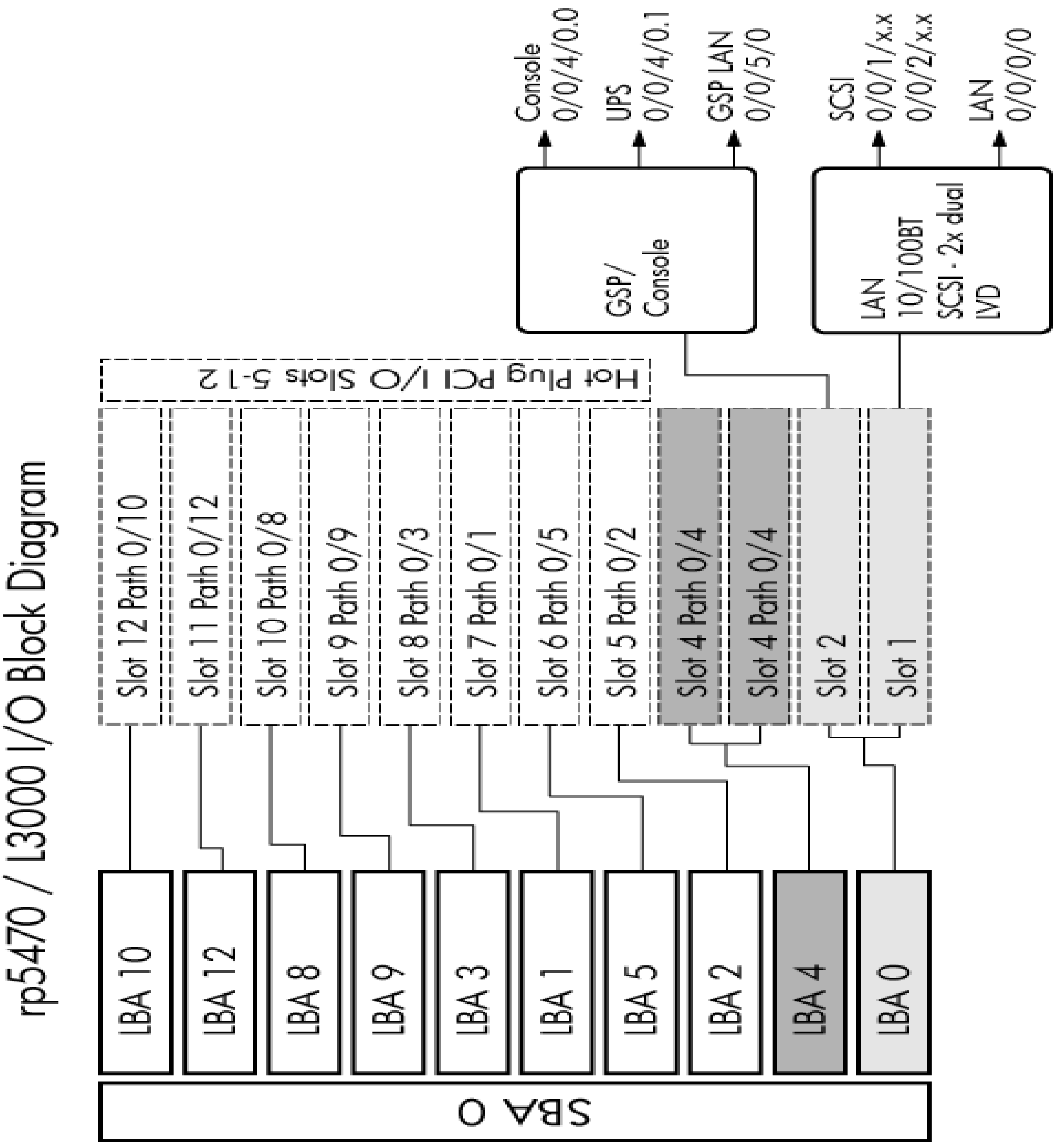
This section contains a simplified PCI IO block diagrams for the rp5470/L3000, rp7400/N4000, and the nPartitionable servers. These diagrams can be used to help determine which LBAs correspond to which physical IO slots.

These were created due the incorrect or inaccessible PA-RISC documentation. For Integrity servers, please see your server manual.

NOTE Note that there is a hardware path change when upgrading from PCI to PCI-X. For more information, see the hardware manuals and “Upgrading Backplanes from PCI to PCI-X” on page 89.

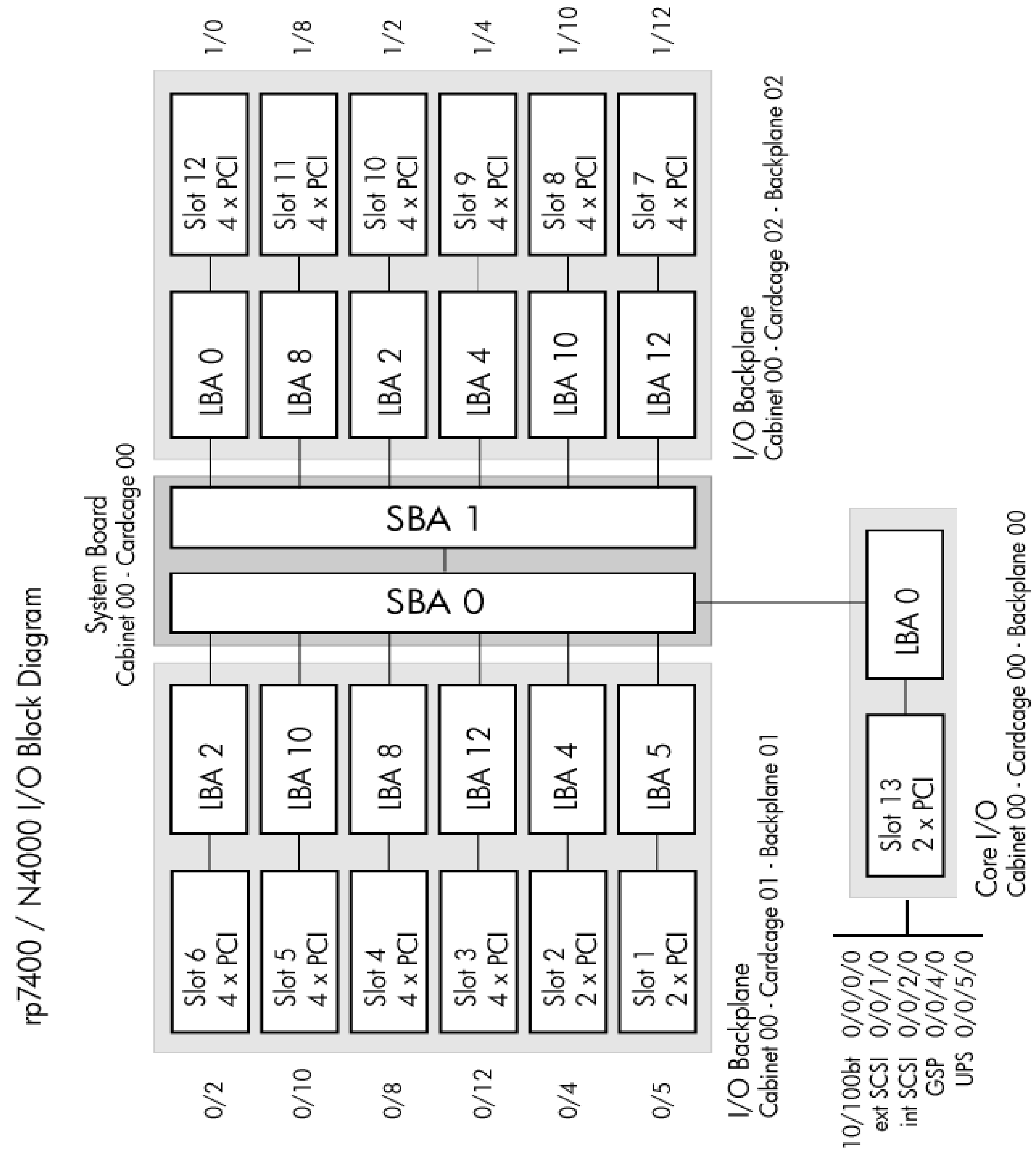
rp5470/L3000 IO Block Diagram

Figure A-1



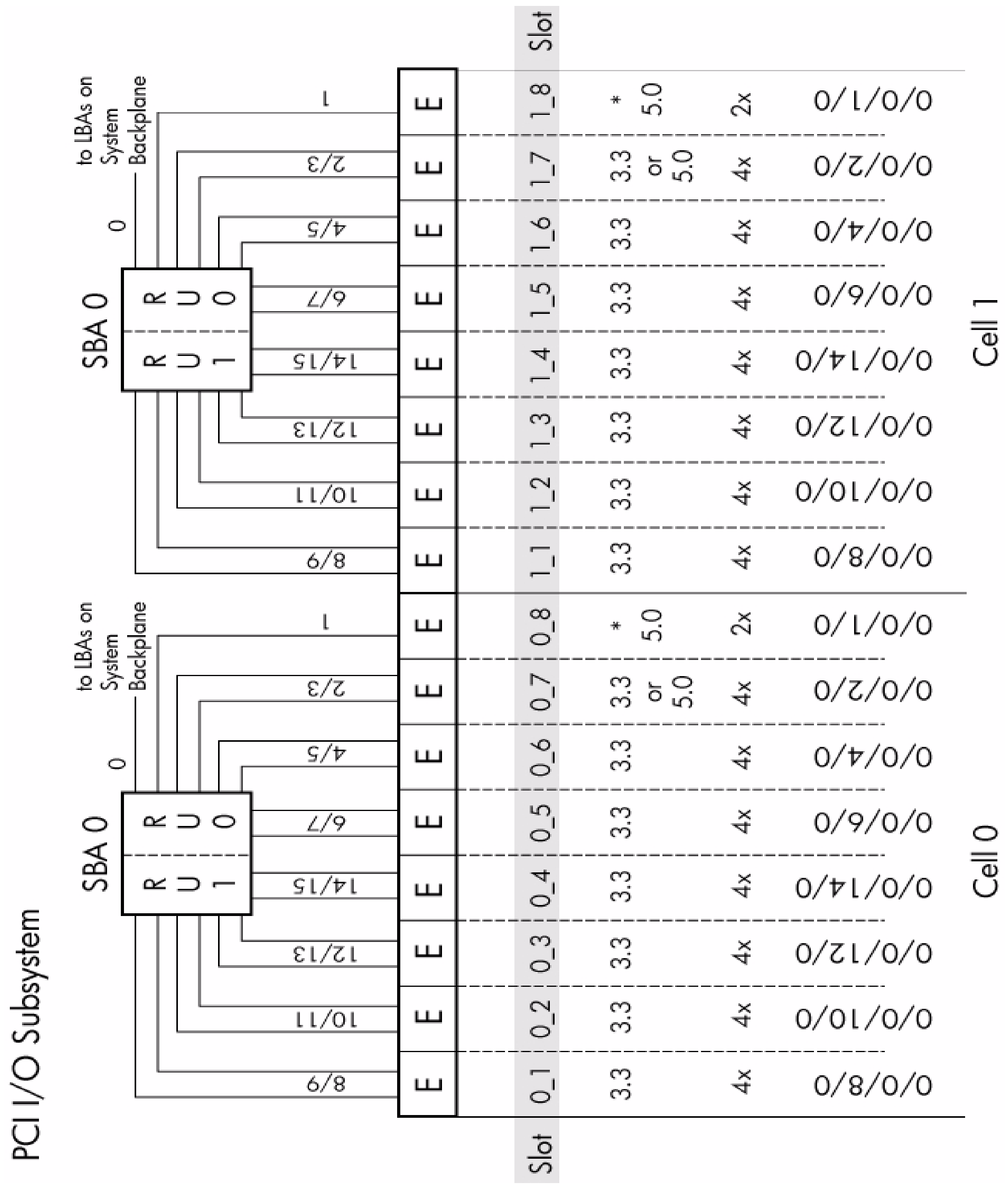
rp7400/N4000 IO Block Diagram

Figure A-2



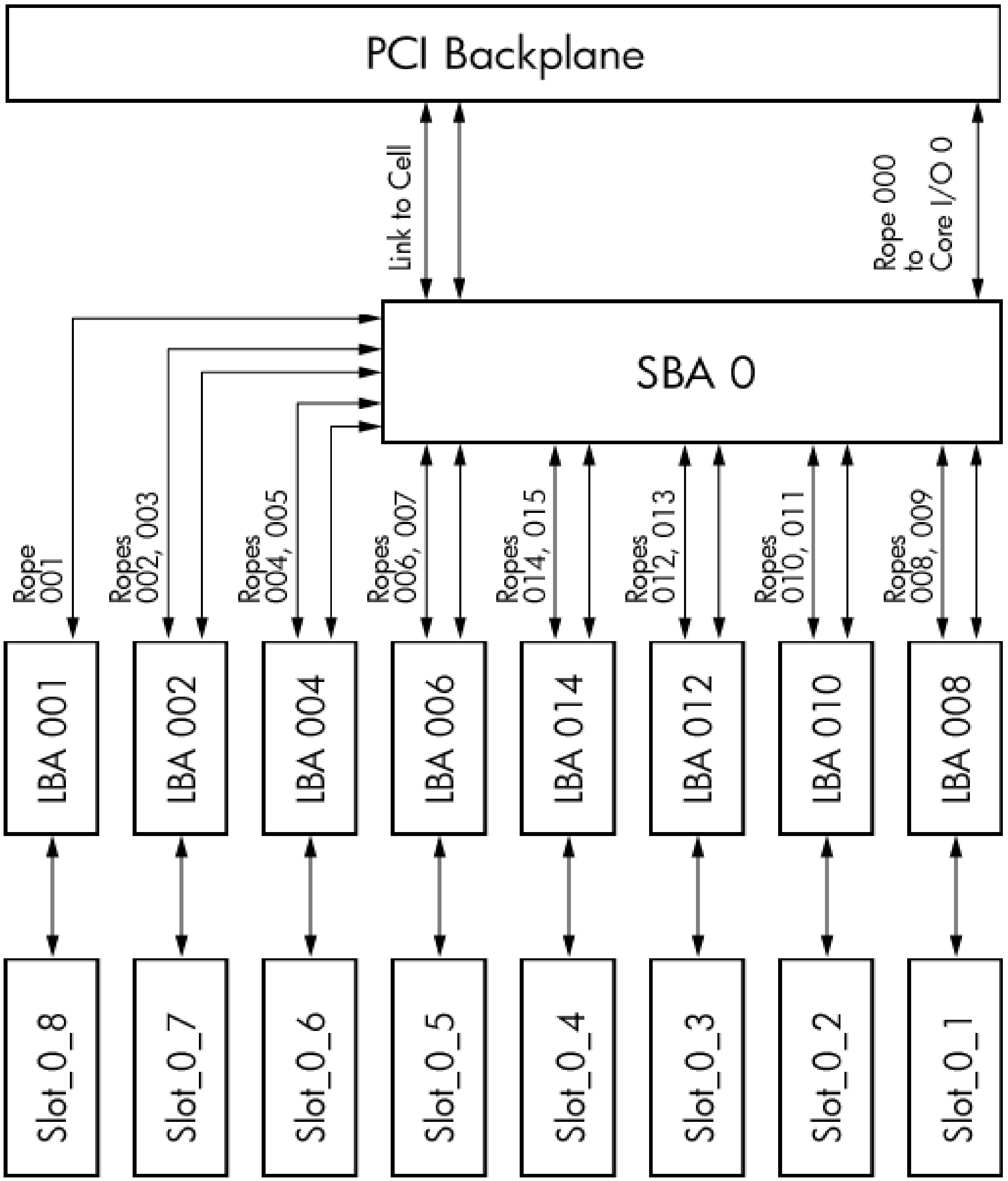
rp7410 and rp7405 PCI IO Block Diagram

Figure A-3



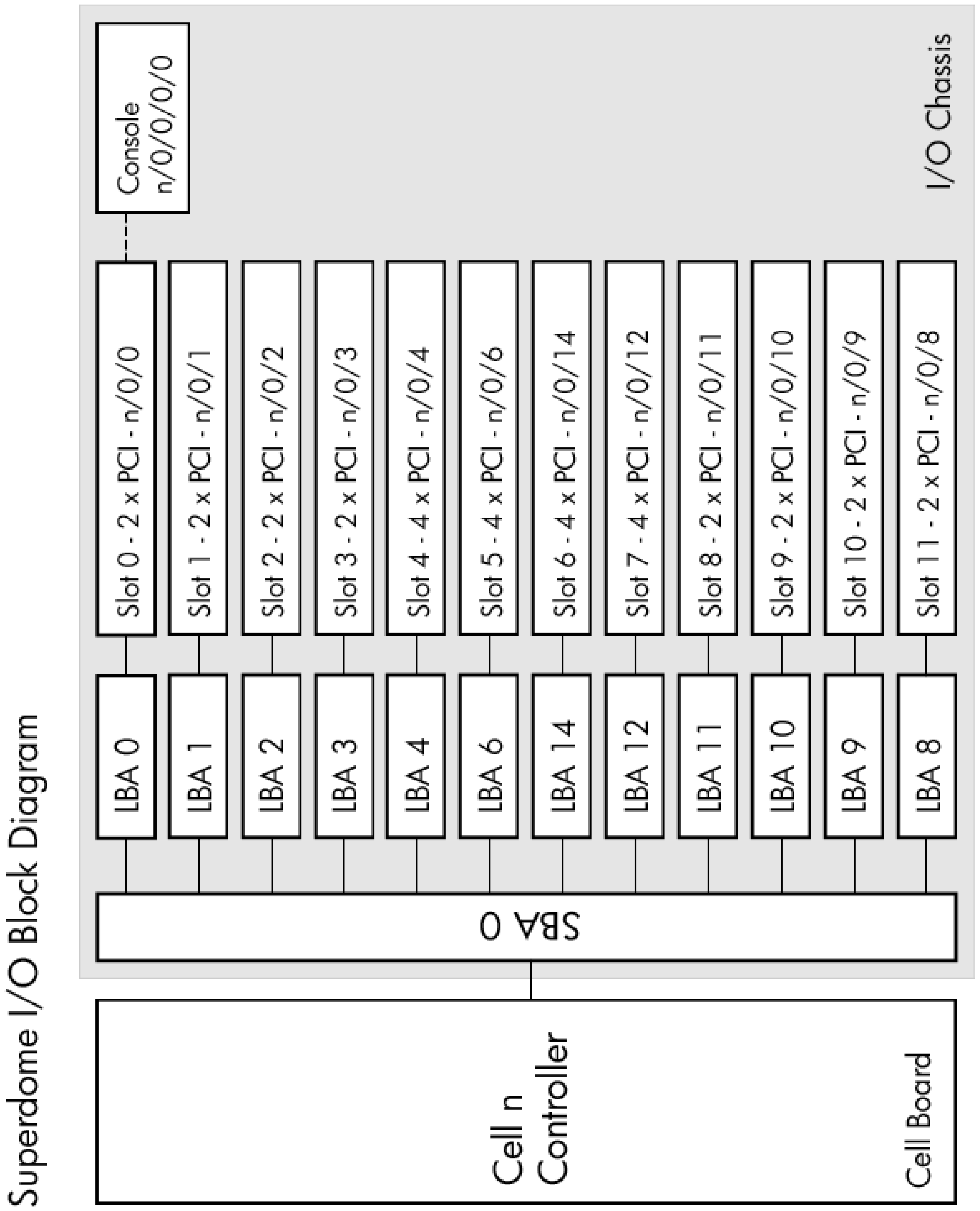
rp8400 PCI IO Block Diagram

Figure A-4



Superdome PCI IO Block Diagram

Figure A-5



B Problem with Adding Unbound CPUs to a Virtual Partition (A.03.xx)

Unbound CPUs allow you to easily adjust processing power between virtual partitions. But a corner case can occur where you will not be able to add specific unbound CPU(s) without rebooting the target partition. This appendix discusses when this situation can occur and how to work around it.

Symptoms

When attempting to add an unbound CPU, you may see the following error message:

```
One or more unbound CPUs were not available when virtual partition <partition_name> was
booted. You must shutdown the partition to add them.
```

This means that the unbound CPU cannot be dynamically added to the virtual partition.

Cause

When a virtual partition boots, the HP-UX kernel creates a table of the existing unbound CPUs available *at the time the virtual partition is booted*. If there is not an existing entry in the table for a specific CPU, that CPU cannot be added to the partition.

Example

To simplify the example, this appendix uses a generic eight-way (8-processor) server whose processors are at the hypothetical hardware paths of x01, x02, x03, x04, x05, x06, x07, and x08.

Create the Virtual Partitions

Suppose we create three partitions using the following commands.

```
# vparcreate -p vpar1 -a cpu::2 -a cpu:::2
# vparcreate -p vpar2 -a cpu::2 -a cpu:::2
# vparcreate -p vpar3 -a cpu::1
```

And suppose that the Monitor chooses the following hardware paths for the bound CPUs:

Virtual Partition	vpar1	vpar2	vpar3
Paths of Bound CPU(s)	x01 x02	x03 x04	x05

This configuration leaves the following three CPUs as unbound CPUs:

Paths of Unbound CPUs	x06, x07, x08

Boot the Virtual Partitions

When we boot the partitions, they will boot with the following bound CPUs; their respective kernels will have the following unbound CPU entries.

Note that the entries for the unbound CPUs are only *entries* for unbound CPUs that can *potentially* be added to the partition. At this point, we have not assigned any unbound CPUs to any of the partitions:

Virtual Partition	vpar1	vpar2	vpar3
Paths of Bound CPU(s)	x01 x02	x03 x04	x05
Unbound CPU Kernel Entries	x06 x07 x08	x06 x07 x08	x06 x07 x08
Paths of Unbound CPUs	x06, x07, x08		

Looking at `vpar3`, because kernel entries for CPUs at `x06`, `x07`, and `x08` exist, any of the unbound CPUs (`x06`, `x07`, or `x08`) *can* be added to `vpar3`. They could also be added to `vpar1` or `vpar2`.

Create A Fourth Virtual Partition

Supposed we create and boot a fourth partition using the following command:

```
# vparcreate -p vpar4 -a cpu::3 -a cpu:::3
```

The Monitor will assign the remaining three CPUs at hardware paths `x06`, `x07`, and `x08`:

Virtual Partition	vpar4
Paths of Bound CPU(s)	x06 x07 x08
Unbound CPU Kernel Entries	(none)

Remove a Virtual Partition

If we shutdown and remove `vpar2` (using `vparremove`), its bound CPUs will become unbound, and the current configuration will be the following:

Virtual Partition	vpar1	vpar3	vpar4
Paths of Bound CPU(s)	x01 x02	x05	x06 x07 x08
Unbound CPU Kernel Entries	x06 x07 x08	x06 x07 x08	(none)
Paths of Unbound CPUs	unbound CPUs are now at `x03` and `x04`		

There are now two unbound CPUs, *but these CPUs are not the same processors that were available at the time the partitions* `vpar1` *or* `vpar3` *were booted.*

Problem is Encountered

At this point, if we attempt to add an unbound CPU to `vpar3` using the following command:

```
# vparmodify -p vpar3 -a cpu::1
```

the command will fail and return the error message:

```
    vparmodify Error: "-a cpu::1": One or more unbound CPUs were not available when virtual
    partition vpar3 was booted. You must shutdown the partition to add them.
```

Although two unbound CPUs are available, their hardware paths are `x03` and `x04`. But the kernel entries for `vpar3` are `x06`, `x07`, and `x08`. Therefore, the command will fail.

The Workaround: Reboot the Target Virtual Partition

Because unbound CPU kernel entries are created when the target partition is booted, you can reboot the target partition so that kernel entries created correctly reflect the available unbound CPUs.

In our example, if we want to add an unbound CPU to `vpar3`, we can reboot `vpar3`:

```
vpar3# vparstatus
vpar3# shutdown -r
```

When `vpar3` boots again, its kernel will create the correct entries for the unbound CPUs, which are now at `x03` and `x04`. The configuration becomes:

Virtual Partition	vpar1	vpar3	vpar4
Paths of Bound CPU(s)	x01 x02	x05	x06 x07 x08
Unbound CPU Kernel Entries	x06 x07 x08	x03 x04	(none)
Paths of Unbound CPUs	unbound CPUs are now at `x03` and `x04`		

Because the unbound CPUs are at `x03` and `x04` and the kernel entries for `vpar3` are `x03` and `x04`, the command to add an unbound CPU to `vpar3`

```
# vparmodify -a vpar3 -a cpu::1
```

now will be successful.

C Calculating the Size of Kernels in Memory (PA-RISC only)

One requirement of vPars is the sum of sizes of the kernels running in memory within a hard partition must be less than 2 GB. This only limits the maximum number of virtual partitions that can be created. If you use the defaults of the dynamic tunables, you will not run into this 2 GB limit. However, if you have adjusted the dynamic tunables, you can perform the calculations described in this appendix to ensure you meet this criteria.

For more information on dynamic tunables, see the white paper *Dynamically Tunable Kernel Parameters in HP-UX 11i* at http://docs.hp.com.

Calculating the Size of a Kernel

To calculate the size of the kernel, perform the following using the kernel file (for example, /stand/vmunix) on the target OS:

Step 1. Get the ending address:

```
# nm /stand/vmunix | grep "ABS|_end"
[10828]  |             212937784|       0|NOTYP|GLOB |0|     ABS|_end
```

The ending address is the second number: `212937784`

Step 2. Get the starting address

```
# nm /stand/vmunix | grep "__text_start$"
[111]    |              201326592|     0|NOTYP|GLOB |0|$FIRST$|__text_start
```

The starting address is the second number: `201326592`

Step 3. Subtract the starting address from the ending address and divide by (1024 * 1024):

```
((212937784 - 201326592) / (1024 * 1024))  + 1 = 12.07
```

Step 4. Round up the result to the next multiple of 64 MB:

```
12.07 rounded to the next multiple of 64 MB = 64 MB
```

Use this number (`64 MB`) for the size of kernel. (Although the actual size of the kernel may be closer to 12 MB, the minimum granularity with which memory is assigned to a partition is 64 MB.)

Examples of Using the Calculations

Changing Dynamic Tunables

If you have already migrated to a vPars server and are adjusting the dynamic tunables of a kernel, check that there is an available memory range under the 2 GB boundary to accommodate the adjusted kernel. You should do this check after adjusting the dynamic tunables but before rebooting the partition.

For example, suppose you calculated the size of an adjusted kernel to be 64 MB. Using `vparstatus -A`, you can check whether there is an available memory range below the 2 GB limit to accommodate the kernel size:

```
# vparstatus -A
...
[Unbound memory (Base  /Range)]:  0x40000000/256
                 (bytes) (MB)
```

The output from `vparstatus -A` shows the following:

- an available `256` MB memory range that can accommodate the 64 MB kernel and
- an available memory range beginning at 0x40000000, which is below the 2 GB limit.

Therefore, the criteria will continue to be met after you reboot the partition.

Migrating OSs from non-vPars Servers to a vPars Server

If you are migrating from multiple non-vPars servers to one vPars server, sum up the results for all the kernels and ensure that the result is under 2 GB.[1]

For example, if we calculated the size of the kernel of the first OS to be 64 MB and the second OS to be 128 MB, the sum is 192 MB. 192 MB is below the 2 GB limit, so we have met the criteria and can migrate the OSs from the multiple non-vPars servers to the single vPars server.

NOTE For vPars A.04 and later, you will need to accommodate for your granularity setting by rounding memory usage to the granularity boundary.

1. Because the Monitor uses 64 MB, the actual number is 1984 MB.

Glossary

bound CPU a CPU that cannot be migrated from or to virtual partitions while the involved virtual partitions are running. Bound CPUs can handle IO interrupts.

dynamic CPU migration the vPars ability to add or remove floater CPUs while a virtual partition is running.

hard partition any isolated hardware environment, such as a rp7400 server or an nPartition within a Superdome complex.

nPartition a logical partition of a complex that divides the complex into groups of cell boards where each group operates independently of other groups. an nPartition can run a single instance of HP-UX or be further divided into virtual partitions.

unbound CPU a CPU that can be migrated from and to virtual partitions while the involved virtual partitions are running. Unbound CPUs cannot handle IO interrupts. Unbound CPUs are also referred to as **floater CPUs**.

virtual consoles a vPars feature that allows console IO to be multiplexed to one hardware console port.

virtual partition software partition of a hard partition where each virtual partition contains an instance of HP-UX. Though a hard partition can contain multiple virtual partitions, the inverse is not true. A virtual partition cannot span a hard partition boundary

vparmgr Virtual Partition Manager. This program provides a GUI to the vPars commands

vPars HP software product that allows software partitioning.

vPars Monitor the program that manages resources using the virtual partition database, boots virtual partitions and their kernels, and emulates certain firmware calls.

vPars partition database the database that contains the configuration for all the virtual partitions.

Index

Index