Managing Hardware Partitions with HP-UX 11i Commands

Shalini Dixit

Keith Johnson

Ching-Ching Katsura

Dennis Mueller

Eugene Wong

Enterprise Systems Technology Laboratory
Hewlett-Packard
19447 Pruneridge Avenue
Cupertino, CA   95014

Eugene Wong
Ph: (408) 447-7164
Fax: (408) 447-6766
Email: eugene_wong@hp.com

HP World

August, 2001

Introduction

Flexible resource management is the key to effective and efficient system utilization. Until recently, system administrators, purchasing agents, and managers were forced to compromise when addressing resource needs that are not constant over time. The Superdome enterprise server supports hardware partitioning, which allows the reconfiguration of processors, memory, and I/O into a variable number of isolated partitions, each running its own HP-UX operating system.

Superdome hardware partitions provide the ability to customize resource allocation as your needs vary over time, but this feature comes at the cost of some increased system administration overhead. The easy-to-use Superdome configuration commands designed to work in concert with Parmgr to specifically aid in the creation, modification, and removal of hardware partitions. These commands are flexible and easy to use. They can be issued from the command line or used in a script.

This paper describes the Superdome hardware partition commands and how they were developed and tested.

What are Partitions?

Before getting into the full background of all the partition commands and how they work, it may be useful to describe some of the common terms and concepts.

Cabinet – all the hardware, which fits into a refrigerator-sized enclosure, consisting of backplane, cells, crossbar connectors, power supplies, fans, and PCI slots.

Cell – a replaceable unit consisting of a circuit board with up to four processors, their memory, controller, bus connectors, and connectors to optional I/O bays.

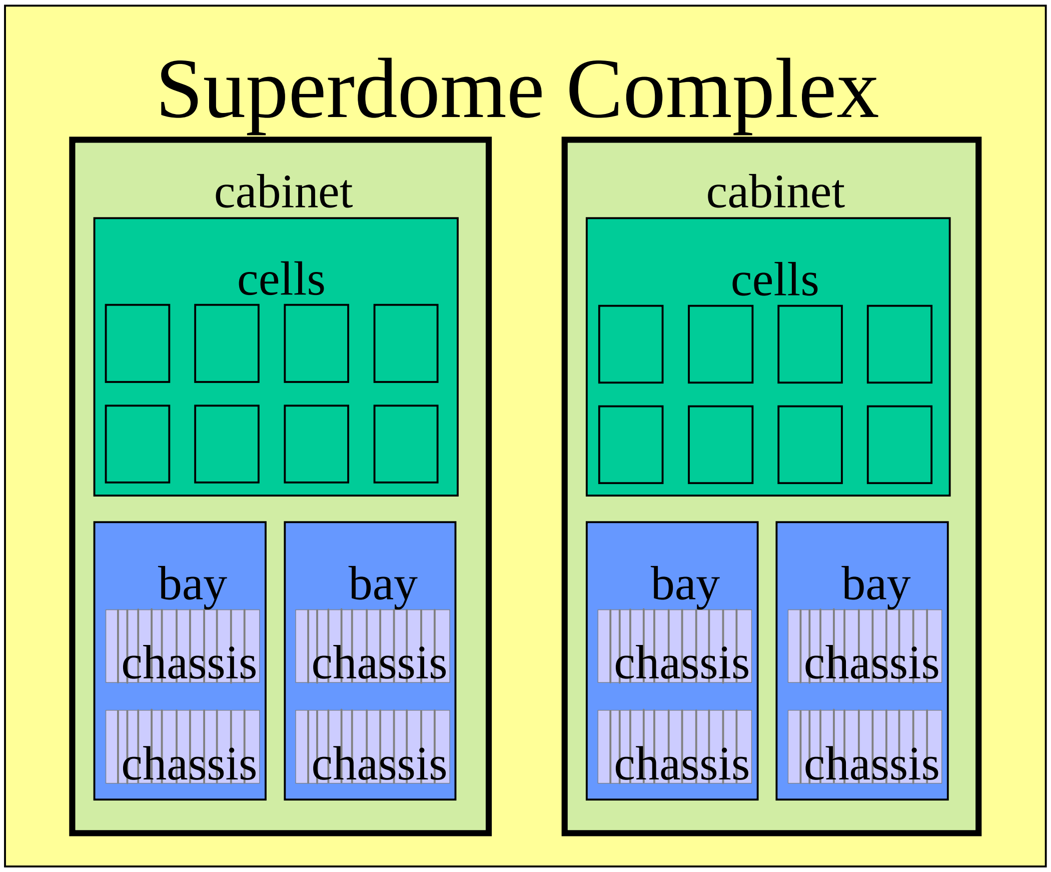
Complex – the total set of hardware components all joined together to form a complete Superdome system. Consists of cabinets containing cells as well as I/O expansion cabinets.

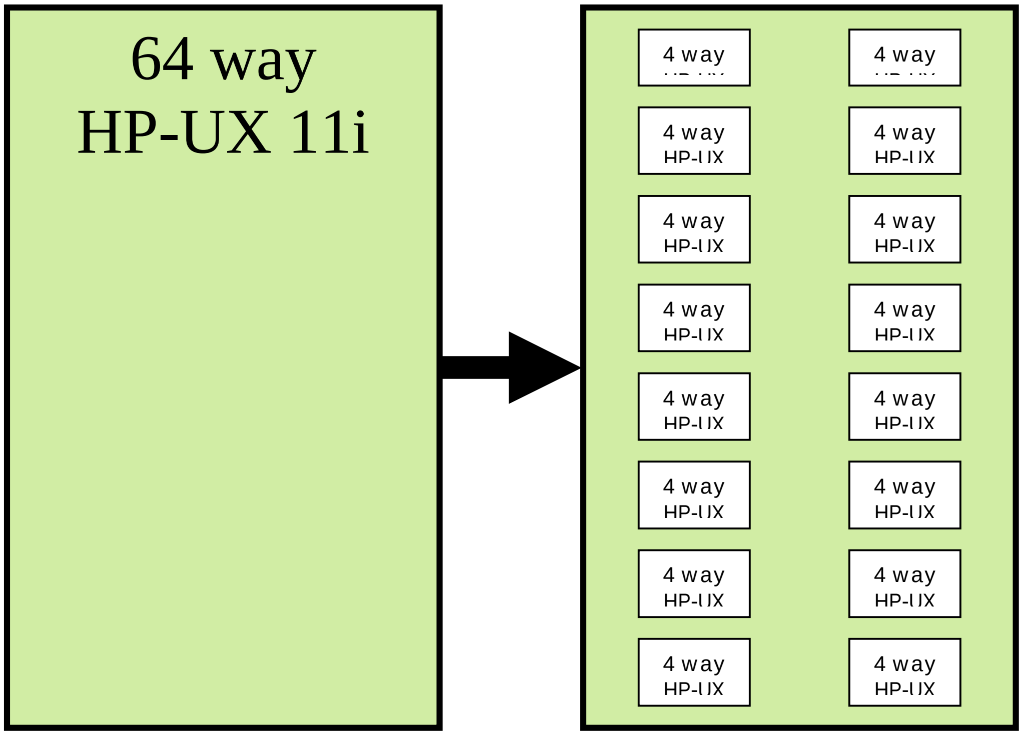
I/O Chassis – an I/O module contained in a Superdome cabinet.

Parmgr – Partition Manager, a GUI interface to the Superdome system administration functions.

Partition – a logical system consisting of cells and core I/O, capable of booting and executing its own operating system separately from other partitions which may also be executing on the same hardware system.

On the following page are two illustrations. The first shows a logical organization of the hardware components of a Superdome system. The second shows how a Superdome system can be converted from a single system into many smaller independent systems, each capable of running its own operating system.


Superdome Complex
cabinet
cells
bay
chassis
chassis
bay
chassis
chassis
cabinet
cells
bay
chassis
chassis
bay
chassis
chassis


64 way
HP-UX 11i
4 way
4 way
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX
4 way HP-UX

Flexible Command-Line Administration

Design of the HP hardware partition commands was based on extensive customer input obtained through contextual inquiries. Teams from HP went to customer sites and asked them to demonstrate how they conducted their system administration tasks. The HP teams watched customer performing these tasks and learned from the customers how they actually perform their system administrations tasks.

The HP teams didn't stop there. They asked customers how we can improve system administration and what they needed in the future. Information from these visits was compiled and used in the design of system administration support for Superdome.

After compiling the customer scenarios, paper prototypes of the proposed partition manager screens were created and presented to the customers in the contextual inquiry studies. They were asked to go through the actions of their daily tasks using the paper prototypes of the Superdome system and the Parmgr in particular. As changes were needed to fit the needs, they were made on the spot with sticky note pages. Additional comments were recorded and evaluated later. Some of the comments from these customer scenarios were very clear: provide both a command line interface in addition to a graphical user interface, provide a way to create scripts for frequently used command sequences, provide detailed and meaningful error messages, and so on.

HP provides two types of system administration tools for the Superdome server: command-line interface based commands and a graphical user interface (Parmgr) which works like the SAM system administration tool.

The GUI provides a convenient and intuitive way to administer hard partitions, however there are many times when it is more practical and efficient to directly use the hard partition commands. It is much more efficient to simply enter a command than to invoke Parmgr to execute a single task.

Many times it is necessary to check the status of a partition or to make simple changes to one or more partition configurations. The commands permit simple updates to the system to be made quickly.

More complex system administration tasks can be put in a script and run all at once. The partition commands have been specifically designed to work well in scripts. Scripts are ideal for periodic modifications to one or more systems since a sequence of commands can be repeated accurately and efficiently. For example, a system's configuration can be changed in the evening to provide more compute power for nightly runs of batch applications and then changed back to the daytime configuration the next morning.

Hardware Partition Commands

The HP hardware partition commands are:

- parstatus
- parcreate
- parmodify
- parremove
- frupower
- fruled

Each command will be described in detail.

parstatus

parstatus command - shows current information about a Superdome complex

Syntax

```
parstatus  [-s]
           [-w]
           [-X]
           [-A] [-M] –C|-I
           [-M] –B|-P
           [-M] –i I/Ochassis [-i]
           [-V|-M] –c cell [-c …]
           [-V|-M] –b cabinet [-b …]
```

Features

- Display entire complex information, including cabinets, cells, chassis, and partitions

```
# parstatus
Warning: No action specified. Default behavior is display all.
[Complex]
   Complex Name : cup2complex
   Complex Capacity
     Compute Cabinet (8 cell capable) : 1
   Active GSP Location : cabinet 0
   Model : 9000/800/SD32000
   Serial Number : USR4001WXY
   Current Product Number : 12345B
   Original Product Number : 12345B
   Complex Profile Revision : 1.0
   The total number of Partitions Present : 3

[Cabinet]
                  Cabinet   I/O       Bulk Power  Backplane
                  Blowers   Fans      Supplies    Power Boards
                  OK/       OK/       OK/         OK/
Cab               Failed/   Failed/   Failed/     Failed/
Num Cabinet Type  N Status  N Status  N Status    N Status       GSP
=== ============  ========= ========= ==========  ============   ======
 0  SD32000        4/ 0/ N+  5/ 0/ ?   5/ 0/ N+    3/ 0/ N+      active
```

```
Notes: N+ = There are one or more spare items (fans/power supplies).
       N  = The number of items meets but does not exceed the need.
       N- = There are insufficient items to meet the need.
       ?  = The adequacy of the cooling system/power supplies is unknown.

[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  4/0/4    2.0/ 0.0 cab0,bay0,chassis1 yes     yes  0
cab0,cell1 active base  2/0/4    2.0/ 0.0 -                  no      yes  2
cab0,cell2 active core  4/0/4    2.0/ 0.0 cab0,bay1,chassis3 yes     yes  1
cab0,cell3 active base  2/0/4    2.0/ 0.0 -                  no      yes  2
cab0,cell4 active core  2/0/4    2.0/ 0.0 cab0,bay0,chassis3 yes     yes  2
cab0,cell5 inactive     4/0/4    2.0/ 0.0 -                  no      -    -
cab0,cell6 inactive     2/0/4    2.0/ 0.0 cab0,bay1,chassis1 yes     -    -
cab0,cell7 active base  2/0/4    2.0/ 0.0 -                  no      yes  2

[Chassis]
                                  Core Connected  Par
Hardware Location   Usage        IO   To         Num
=================== ============ ==== ========== ===
cab0,bay0,chassis0  absent       -    -          -
cab0,bay0,chassis1  active       yes  cab0,cell0 0
cab0,bay0,chassis2  absent       -    -          -
cab0,bay0,chassis3  active       yes  cab0,cell4 2
cab0,bay1,chassis0  absent       -    -          -
cab0,bay1,chassis1  inactive     yes  cab0,cell6 -
cab0,bay1,chassis2  absent       -    -          -
cab0,bay1,chassis3  active       yes  cab0,cell2 1

[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ==============================
 0  active         1      1     cab0,cell0
 1  active         1      1     cab0,cell2 partition1
 2  active         4      1     cab0,cell4 partition2
```

-Display all partitions, showing active/inactive, assigned cell count, assigned chassis count, partition name

```
# parstatus -P
[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ==============================
0  active         1      1     cab0,cell0
1  active         1      1     cab0,cell2 partition1
2  active         4      1     cab0,cell4 partition2
```

-Display all cell information including cell usage, partition assignment, connected chassis. parstatus also shows intermediate and self-test states when a cell is temporarily unavailable.

```
# parstatus –C
[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  4/0/4    2.0/ 0.0 cab0,bay0,chassis1  yes     yes  0
cab0,cell1 active base  2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell2 active core  4/0/4    2.0/ 0.0 cab0,bay1,chassis3  yes     yes  1
cab0,cell3 active base  2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell4 active core  2/0/4    2.0/ 0.0 cab0,bay0,chassis3  yes     yes  2
cab0,cell5 inactive     4/0/4    2.0/ 0.0 -                   no      -    -
cab0,cell6 inactive     2/0/4    2.0/ 0.0 cab0,bay1,chassis1  yes     -    -
cab0,cell7 active base  2/0/4    2.0/ 0.0 -                   no      yes  2
```

-Display single partition in detail, showing firmware levels, boot paths, chassis assignments, and partition name

```
# parstatus -V -p 2
[Partition]
Partition Number       : 2
Partition Name         : partition2
Status                 : active
IP address             : 0.0.0.0
Primary Boot Path      : 0/0/0/0/0.0.0
Alternate Boot Path    : 0/0/0/0/0.0.0
HA Alternate Boot Path : 0/0/0/0/0.0.0
PDC Revision           : 10.0
IODCH Version          : 23664
CPU Speed              : 552 MHz
Core Cell              : cab0,cell4

[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,cell1 active base  2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell3 active base  2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell4 active core  2/0/4    2.0/ 0.0 cab0,bay0,chassis3  yes     yes  2
cab0,cell7 active base  2/0/4    2.0/ 0.0 -                   no      yes  2

[Chassis]
                                  Core Connected  Par
Hardware Location   Usage         IO   To         Num
=================== ============ ==== ========== ===
cab0,bay0,chassis3  active       yes  cab0,cell4 2
```

-Display all I/O chassis, show usage, power status, connected cell, partition assignment

```
# parstatus -I
[Chassis]
                                  Core Connected  Par
Hardware Location   Usage        IO   To         Num
=================== ============ ==== ========== ===
cab0,bay0,chassis0  absent       -    -          -
cab0,bay0,chassis2  absent       -    -          -
cab0,bay1,chassis0  absent       -    -          -
cab0,bay1,chassis1  inactive     yes  cab0,cell6 –
cab0,bay1,chassis2  absent       -    -          -
```

-Display single I/O chassis

```
# parstatus -i 0/1/1
[Chassis]
                                  Core Connected  Par
Hardware Location   Usage        IO   To         Num
=================== ============ ==== ========== ===
cab0,bay1,chassis1  inactive     yes  cab0,cell6 -
```

-Display cabinet 0, showing power supply and fan status

```
# parstatus -V -b 0
 [Cabinet]
                  Cabinet   I/O       Bulk Power  Backplane
                   Blowers   Fans      Supplies    Power Boards
                  OK/       OK/       OK/         OK/
 Cab              Failed/   Failed/   Failed/     Failed/
 Num Cabinet Type  N Status  N Status  N Status    N Status       GSP
 === ============  ========= ========= ==========  ============   ======
  0  SD32000        4/ 0/ N+  5/ 0/ ?   5/ 0/ N+    3/ 0/ N+      active

 Cabinet Blowers
 ===============
 Fan 0  ok
 Fan 1  ok
 Fan 2  ok
 Fan 3  ok

 I/O Fans
 ==============
 Fan 0  ok
 Fan 1  ok
 Fan 2  ok
 Fan 3  ok
 Fan 4  ok

 Bulk Power Supplies(BPS)
 ========================
 Power Supply  0  ok
 Power Supply  1  ok
 Power Supply  2  ok
 Power Supply  3  ok
 Power Supply  4  ok
```

```
Backplane Power Boards
======================
Power Supply  0  ok
Power Supply  1  ok
Power Supply  2  ok


Notes: N+ = There are one or more spare items (fans/power supplies).
       N  = The number of items meets but does not exceed the need.
       N- = There are insufficient items to meet the need.
       ?  = The adequacy of the cooling system/power supplies is unknown.
```

-Display the number of the partition in which the command is currently running

```
# parstatus -w
The local partition number is 0.
```

parcreate

parcreate command - creates a new Superdome partition

Syntax

parcreate –P Partition Name –I IPaddress –c cell:[cellType]:[use_on_next_boot]:[failure_usage] [-c …] [-b path] [-t path] [-s path] [-r cell] [-r …] [-B] [-k s_lock]

Features

The user specifies:

- a list of free cells to include in the partition
- whether to use each cell in the next boot
- partition boot path(s)
- root cell alternates
- whether to boot this partition immediately (optional)

Root permission is required.

```
# parstatus -C     before command

# parcreate -c 4::: -c5::: -c 6::: -c 7::: -b 4/0/1/0/0.12 -B

# parstatus -C     after command
```

parmodify

parmodify command - modifies an existing partition

Syntax

parmodify –p PartitionNumber

```
{-a cell:[cellType]:[use_on_next_boot]:[failure_usage] [-a …] |
  -m cell:[cellType]:[use_on_next_boot]:[failure_usage] [-m …] |
   -I IPaddress | -r cell [-r …] | –d cell [-d …] |
   -b path | -t path | -s path | -P PartitionName | -B |-k s_lock:p_lock}
```

Features

The user specifies one or more of the following:

- partition number (required)
- cell number to add, modify, or delete
- use on next boot flag if add or modify
- root cell alternate changes
- boot path changes
- new partition name
  - whether to boot this partition immediately
  -

Root permission is required

```
# parstatus -C before command showing inactive partition with 1 cell

# parmodify -p 1 -a 3::: -P "cells 1 and 3"

# parstatus -C after command

# parstatus -p 1 after command
```

parremove

parremove command - removes a partition

Syntax

parremove –p PartitionNumber [-F] [-k s_lock:p_lock]

Features

The user specifies:

- partition number
- whether to force removal of the current active partition from within

The parremove command can remove any inactive partition or the current active partition. Removal of an active partition does not take place until it has been shut down. Root permission is required.

```
# parstatus -P before command

# parremove -p 1
# parstatus -P after command
```

frupower

frupower command - power on/off cells or I/O chassis; display power status

Syntax

```
frupower [-d | -o | -f] –c cell [-c …]
         [-d | -o | -f] –i I/Ochassis [-i …]
         [-d] –C [-l cabinet] [-l …]
         [-d] –I [-l cabinet] [-l …]
```

Features

The user supplies one or more of the following:

Actions - off, on or display
List all components – cells, I/O chassis
Power on all cells
Power on all I/O chassis
Limit power on all items to a specified cabinet or cabinets

Root permission is required.

```
# frupower -d -C (shows power status of all cells)

# frupower -f -c 1 (power off cell 1)

# frupower -d -c 1  (display power status of cell 1)

# frupower -d –C   (display power status of all cells)
```

fruled

fruled command - flash/turn off attention LED for cell, I/O chassis, or cabinet

Syntax

```
fruled [-f | -o] [-B] –c cell [-c …]
       [-f | -o] [-B] –i I/O chassis [-i …]
       [-f | -o] –b cabinet [-b …]
       [-f] –C [-l cabinet] [-l …]
       [-f] –I [-l cabinet] [-l …]
```

Features

The user supplies one or more of the following:

Actions – flashing on or off
Components - cell, chassis, or cabinet LED
Turn off all cells or chassis LEDs in a selected cabinet
Turn off all cells or chassis in cabinet where selected cell or chassis resides

```
# fruled -o -c 1   # flashing LED indicator for cell 1
```

An Example Scenario

The following example is the scenario of a system administrator who wants to reconfigure a Superdome to include additional partitions. The system is currently in use and there is already a list of components that are known to be available. For this example, assume the existence of additional cells 1 through 4, and that cells 2 and 4 are core cells.

1. We start by taking a look at the system's initial configuration to find that there is only one partition defined and active. It consists of a single cell with a minimum of one I/O chassis connected to it (also called a core cell). It does not have a partition name.

```
# parstatus -P
[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ==============================
 0  active         1      1     cab0,cell0
```

2. To create a new partition with core cell 2 is a very straightforward command.

```
# parcreate -P partition1 -c 2:::
Partition Created. The partition number is: 1
```

3. Create a new partition named partition2, containing core cell 4 and non-core cells 1 and 3.

```
# parcreate -P partition2 -c 1::: -c 3::: -c 4:::
Partition Created. The partition number is: 2
```

4. At this point we want to install a new cell board into slot 7. Using the fruled command, we can light up the correct slot to show where to insert the cell board.

```
# fruled -o -c 7
```

5. Once the cell board has been installed, it can be powered up so that it can be used.

```
# frupower -o -c 7
```

6. Now the cell is ready to be added to partition 2 using the parmodify command.

```
# parmodify -p 2 -a 7:::
Command succeeded.
```

7. Now the parstatus command can be used to give a summary of all the existing partitions.

```
# parstatus -P
[Partition]
Par              # of  # of I/O
Num Status       Cells Chassis  Core cell  Partition Name (first 30 chars)
=== ============ ===== ======== ========== ==============================
 0  active         1      1     cab0,cell0
 1  inactive       1      1     ?          partition1
 2  inactive       4      1     ?          partition2
```

8. For a detailed listing showing all the components of the partitions and their status, use the parstatus command with the option to show the entire complex.

```
# parstatus -C
[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  4/0/4    2.0/ 0.0 cab0,bay0,chassis1  yes     yes  0
cab0,cell1 inactive     2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell2 inactive     4/0/4    2.0/ 0.0 cab0,bay1,chassis3  yes     yes  1
cab0,cell3 inactive     2/0/4    2.0/ 0.0 -                   no      yes  2
cab0,cell4 inactive     2/0/4    2.0/ 0.0 cab0,bay0,chassis3  yes     yes  2
cab0,cell5 inactive     4/0/4    2.0/ 0.0 -                   no      -    -
cab0,cell6 inactive     2/0/4    2.0/ 0.0 cab0,bay1,chassis1  yes     -    -
cab0,cell7 inactive     2/0/4    2.0/ 0.0 -                   no      yes  2
```

Commands Development

All of the commands were developed using the principles of Personal Software Process (PSP). Great emphasis was placed on peer reviews at all process steps and defect prevention in general. The code was designed to be simple and yet perform all the complex tasks. The function modularity and shared code make these commands easy to maintain.

As part of the development process, there were extensive reviews of all design documents, test plans, as well as code. The external specifications were created shortly after a detailed investigation of the user needs. These specifications were formalized into the man pages. Then the high level and low level design specifications were created, simultaneously with the test plan. Each of these specifications was reviewed by the entire team prior to the development of code.

Test matrices were designed to ensure coverage for all the functional features of the Superdome partitioning commands. The matrices helped in organizing the development of the tests needed to verify all of the functionality. These measures, along with the use of C-Cover, a tool for measuring code coverage, were extremely valuable in ensuring adequate testing.

The use of common code modules in developing the commands also helped in reducing the complexity and simplifying maintenance. The common code modules were primarily used for user input verification, error handling, and access to key system functions.

Commands Testing

Commands testing was predicated on two metrics: 100% functional coverage and at least 85% path-flow coverage. In order to ensure 100% functional coverage, matrices were developed to map test cases to command functions. Command functionality was divided into two categories: command interface (API) and command operation.

Development of the command interface matrices resulted in the identification of "negative" tests; test cases that are suppose to fail. For example, inputting a partition name containing invalid characters would result in an intentionally negative test case. Another example would be attempting to remove a non-existing partition.

The command operation test cases were "positive" tests. They exercised functionality that impacted or reported the state of a partition or complex. There were also negative tests which attempted to execute prohibited operations or tried to use out-of-bounds values on the commands.

Both the test engineers and the command development engineers subjected the test coverage matrices to intensive inspection. These inspections ensured that the matrices, and the resulting test cases, fully covered 100% of the possible command options and their functionality. This approach ensured full "black box" test coverage.

Path-flow analysis augmented the functional testing and provided "white box" coverage by exercising at least 85% of all code paths. C-Cover was used to analyze this coverage and to identify additional test cases to be developed and implemented. We used three functional simulators to aid in development and testing. These simulators allowed us to exercise code paths that would not have been exercised in normal command usage.

We selected the goal of 85% path-flow coverage as an ambitious objective. In any well-designed code there are a number of paths that are designed to respond to error conditions which, in all probability, will never occur but which must be included in order to guarantee completeness of design. The 85% goal for path-flow coverage proved to be a reasonable target for ensuring adequate test coverage.

This two-pronged approach of functional coverage verified by test matrices, coupled with path-flow analysis enabled the team to fully test the hardware partition commands and verify the high quality level of our code.

Use of Simulators

The commands were developed concurrently with the Superdome hardware, firmware, and the HDCI library development. The commands developers needed a test-bed for their code before the various parts of the system could be brought together. We made use of three software simulators during the development and testing of the HP hardware partition commands:

1. an architecture simulator which was used in the early phases to simulate the Superdome hardware instruction set and interpreted all the instructions in the HP-UX kernel and commands,
2. an HDCI simulator that we wrote to simulate the configuration interface which presents the programmatic interface to the firmware and hardware states to the commands, and
3. a simulator of the system administration GUI that was a later replacement for the HDCI simulator.

In the early stages of development, use of simulators gave the code developers an alternative to waiting for the hardware and firmware to be available for testing. An extensive Superdome architectural simulator had been developed to simulate the system firmware and low-level programmable hardware. It was used during the development, and test of Superdome firmware and low-level kernel software. This simulator was also useful to the command developers prior to the availability of a usable functioning hardware/software stack, although it was never intended to simulate a Superdome viewed from user space and lacked the ability to simulate more than one partition at a time.

Since the architectural simulator provided emulation of the computer instruction set, it worked very slowly on something as large as the entire HP-UX system. It was very useful in debugging the instruction set or small routines, but was largely impractical for our purposes since we needed the entire kernel and I/O operations and it took far too much time to emulate all the instructions in the kernel. So the HDCI simulator was a much-needed tool to allow us to check the flow of logic in the commands prior to getting access to the real hardware and HDCI software.

The HDCI functions represent elements such as cells, partitions, and Superdome cabinets in various data structures. These structures can be simulated, but they must be stored between command invocations. A simulator was written that mimicked HDCI functions and maintained the data structures in local data files. It provided sufficient emulation of the HDCI for the commands to create and manipulate partitions at the data structure level. The simulator also allowed a head start on test development. A more sophisticated form of the HDCI simulator remains in use for regression testing. Both HDCI simulators support multiple Superdome configurations and both provide ways to insert hardware and software failure modes that cannot be created on demand on a real system.

The HDCI simulator was extremely useful during the testing stages as we tried to test all the execution paths. The HDCI simulator had a feature, which allowed error returns to be taken for each of the low-level function calls. This allowed many of the error handling paths to be tested as we did the path-flow analysis.

Summary

This discussion has shown that hardware partitions provide a valuable tool for resource management. Customers now have the ability to easily reconfigure resource allocation as their requirements change, without having expensive hardware sit idle during low-demand times, and while being able to run multiple or different versions of HP-UX on the same platform.

This flexibility is enabled by the HP hard partition commands. These commands were designed from well-researched customer requirements, and were developed and tested using proven methods ensuring conformity to very high quality standards.

HP hard partitions, and their system administration commands, are valuable additions to HP’s extensive line of enterprise servers.