

# Configuring and Migrating Memory on vPars

A Technical White Paper



Introduction .....	2
Memory Categories .....	3
Interleaved and Cell Local Memory .....	3
ILM and CLM Guidelines.....	5
Base and Floating Memory.....	6
Base and Floating Memory Guidelines.....	8
Memory Granules – Containers for Physical Memory.....	10
Memory Granules Guidelines.....	11
Memory Resource Assignment.....	13
Memory Allocation – Non-live Database .....	13
Memory Allocation and Binding – Live Database.....	14
User Specified Ranges.....	16
Memory Migration Management.....	18
Tools to Determine Memory Usage.....	18
Monitoring the Status of an Online Operation.....	19
Cancelling an Online Operation.....	19
Memory Migration Example.....	19
Experimental Setup .....	20
Memory Usage on vpar1 .....	20
Memory Usage on vpar2.....	21
Deletion of Memory from vpar1 .....	21
Addition of the Freed Memory to vpar2.....	23
Memory Accounting.....	25
References.....	28

# Introduction

Virtual Partitions (vPars) is a feature on HP-UX 11i based systems that allows a system administrator to divide the hardware resources on a single L-class, N-class, and cellular-based hard partition into one or more logical partitions. This is accomplished through a software layer called the vPars monitor that resides between the operating system kernel and the firmware. The vPars monitor controls the ownership of the processors, memory, and I/O resources on the system. For each partition to be minimally operational, it must contain at least one processor, enough memory to boot and run HP-UX, and one or more I/O cards connected to storage and LAN. Each partition runs its own copy of the operating system thereby providing namespace and software fault isolation.

The system administrator can specify the resources for each partition using a command-level interface. This partitioning information is stored in a file and that file is commonly referred to as the vPars database. During system boot, the vPars monitor takes inventory of the resources on the system, reads the vPars database, and assigns the available resources to the defined partitions according to the partition plan stored in the vPars database. When each partition is launched, the operating system calls firmware to discover the subset of the system resources it owns. The vPars monitor, with help from firmware, exposes to each partition's kernel only those processors, memory, and I/O cards that the partition owns. Hence, each partition only accesses the subset of resources on the system that it owns.

The main objective of this document is to describe memory resource configuration in a vPars system with particular emphasis on memory migration; that is, the ability to dynamically add and delete memory from partitions. The document describes the various choices available to the user to control memory assignment to each partition with generic guidelines to follow while using these choices, the binding of physical memory done by the vPars monitor and the tools available to observe and validate memory assignment.

The document is intended for system administrators and users of all vPars releases who configure and manage memory. While perusing this document, users of vPars A.04.xx release can skip the discussion related to memory migration and users of vPars A.01.xx, A.02.xx and A.03.xx releases can skip locality and memory migration discussions.

The rest of this document is divided into six chapters:

- The "[Memory Categories](#)" chapter describes different memory types available for assignment.
- The "[Memory Granules – Containers of Physical Memory](#)" chapter describes the granules, which are the unit of memory resources available for assignment.
- The "[Memory Resource Assignment](#)" chapter describes how monitor assigns memory to each partition.
- The "[User Specified Ranges](#)" chapter describes how user can explicitly control the memory binding.
- The "[Memory Migration Management](#)" chapter describes the tools available to manage memory migration.
- The "[Memory Accounting](#)" chapter details the commands available to the user to check memory assignment.

## Memory Categories

The vPars A.01.xx, A.02.xx and A.03.xx release streams support partitioning on HP-UX 11i v1 systems which only run on uniform memory systems (interleaved memory on cellular systems). The vPars A.04.xx release stream supports partitioning on HP-UX 11i v2 systems which support memory locality based optimizations on non-uniform memory access (NUMA) systems. The vPars A.05.xx release supports partitioning on HP-UX 11i v3 systems which allow online addition and deletion of memory. In these systems, the locality of the memory or the memory that can be online deleted later has to be identified as such to the kernel when the partition is booted or when memory is added. Hence, to allow users to control the type of memory that will be part of the partition, vPars supports the following memory categories: interleaved memory (ILM), cell local memory (CLM), base memory, and floating memory. The next section details ILM and CLM. The section after that explains base and floating memory.

### Interleaved and Cell Local Memory

HP-UX 11i v2 and later releases support memory locality based optimizations on NUMA systems. In these systems the administrator can carve out portions of memory from one or more cells as cell local memory. The remaining memory from all the cells that is not cell local memory is interleaved at cache line granularity by firmware during system boot to create interleaved memory with average access latency. The kernel creates one locality for each memory type with differing access latency, transparently optimizes placement of data to minimize memory latency and provides APIs [1] to applications to control the locality they want to use. The applications can take advantage of this by using processors and memory from the same localities or localities that are close to each other to achieve better performance.

In A.04.xx and A.05.xx vPars system, the system administrator has a choice of assigning ILM as well as CLM between partitions. The vPars commands have separate options for ILM and CLM. The legacy A.01.xx, A.02.xx and A.03.xx release vPars memory options would be considered as ILM. In order for the user to specify the locality while performing memory operations with the `vparcreate` or `vparmodify` commands, these commands have an additional option that takes the cell number as an argument when specifying memory sizes. This option also provides the ability to tie processors from the same locality to the partition. With a cell number, the memory and processors are considered as cell local memory and cell local processors (CLP) to be allocated from the specified cell.

The system administrator has two choices: configure all the memory as ILM and manage it similar to the way it was managed in A.01.xx, A.02.xx and A.03.xx release vPars systems or configure both ILM and CLM and divide it among partitions. The following lists some of the steps that the system administrator can use to create and use ILM and CLM in the vPars system:

1. Take inventory of the amount of ILM and CLM in each cell using the `parstatus` command.

```
# parstatus -p <hard partition number> -V
and
# parstatus -c <cell number> ... -V
```

2. Decide on the number of partitions, the amount of ILM for each partition, and the amount of CLM for each partition and the cell or cells from which to allocate CLM.
3. Use `parmodify` to carve out CLM from cells. Cross-check through the `parstatus` command that the required amount of CLM has been requested.

```
# parmodify -p <nPar > -m <cell number>:::<percentage CLM>% ...
```

4. Create each partition using `vparcreate` and assign the required amount of ILM and CLM to the partition using the memory options of the `vparcreate` and `vparmodify` commands.

```
# vparcreate -p <vPar> -a mem::<ilm> -a cell:<num>:mem::<clm> ...  
Or  
# vparmodify -p <vPar> -a mem::<ilm> -a cell:<num>:mem::<clm> ...
```

5. If needed, assign CLP to have CPUs and memory from the same locality using the `vparcreate` and `vparmodify` commands.

```
# vparcreate -p <vPar> -a cell:<num>:cpu::<clp> ...  
Or  
# vparmodify -p <vPar> -a cell:<num>:cpu::<clp> ...
```

6. Reboot the system and boot the monitor using the newly created database.

In order to illustrate how the memory would be divided up as ILM and CLM, we will consider setting up a sample vPars system with two partitions: `vpar1` and `vpar2`. The example system contains two cells: cell 0 and cell 1. Each cell contains 4 GB of memory. The following steps illustrate how ILM and CLM are created in the system and distributed among `vpar1` and `vpar2`.

First, we need to get an idea of how much memory is available on the system. Invoking the `parstatus` command with `parstatus -p 0 -v` and `parstatus -c 0 -c 1 -v` shows something like the following for memory related output (the rest of the output is not included here):

```
...  
Total Good Memory Size : 8.0 GB  
Total Interleave Memory: 4.0 GB  
...
```

And

```
...  
Global Cell Number      : 0  
...  
Requested CLM value     : 2.0 GB  
Allocated CLM value     : 2.0 GB  
...  
Memory OK               : 4.00 GB  
...  
Global Cell Number      : 1  
...  
Requested CLM value     : 2.0 GB  
Allocated CLM value     : 2.0 GB  
...  
Memory OK               : 4.00 GB
```

The system administrator has already set up 50% of the memory from each cell to be CLM. Hence, the system contains 2 GB of CLM from cell 0 and 2 GB of CLM from cell 1, and 2 GB of memory from cell 0 is interleaved with 2 GB memory from cell 1 to yield 4 GB of total ILM. Now, in order to create the two partitions with 1 GB and 2 GB of ILM respectively, we would invoke the following `vparcreate` commands:

```
# vparcreate -p vpar1 -a mem::10241  
# vparcreate -p vpar2 -a mem::2048
```

---

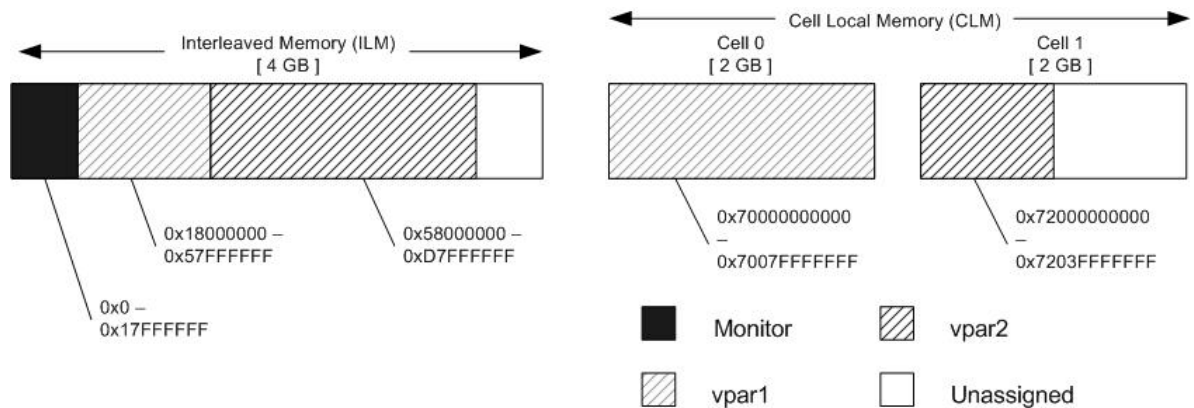
<sup>1</sup> The examples used in this document might assign 512 MB or 1024 MB of memory while creating partitions. The actual amount of memory that a partition requires depends on the HP-UX release that partition will contain. Hence, system administrators should refer to the HP-UX release documentation to determine the minimum amount of memory required while creating or configuring the partitions.

Note that in the above examples, we have merely specified a memory size without any particular cell information. Hence the memory is allocated from the ILM present on the system. Now, if we need to add 2 GB of CLM from cell 0 to vpar1 and 1 GB of CLM from cell 1 to vpar2 and also add 2 CPUs from cell 0 to vpar1 and 2 CPUs from cell 1 to vpar2, we would do the following:

```
# vparmodify -p vpar1 -a cell:0:mem::2048 -a cell:0:cpu::2
# vparmodify -p vpar2 -a cell:1:mem::1024 -a cell:1:cpu::2
```

Figure 1 below depicts this memory distribution for the above example on an HP Integrity server when both vpar1 and vpar2 are live. The low end of the memory is at the left and the high end is at the right in the figure. The system contains 4 GB of ILM starting from address 0, 2 GB of CLM from cell 0 starting from address 0x7000000000 and 2 GB of CLM from cell 1 starting from address 0x7200000000. Approximately 384 MB from address 0 is taken by the monitor and firmware on this example system. Note: Memory consumed by the monitor and firmware can vary.

Figure 1 ILM and CLM Partitioning



### ILM and CLM Guidelines

The following provides some guidelines on how to use ILM and CLM in a vPars system:

- If ease of management is the objective, configure all the memory as ILM unless one or more partitions in the system are going to use applications that make use of locality APIs to optimize performance. Having all memory as ILM makes it easier to assign and move memory and processors between partitions.
- When configuring CLM on the system, make sure that there is enough ILM reserved for each partition to load and run the kernel. The PA-RISC HP-UX kernel needs ILM to boot. Even though on the HP Integrity server, HP-UX kernel might boot with just CLM, it is not a supported HP-UX configuration.
- When configuring CLM and CLP for each partition, refer to the HP-UX NUMA documentation to decide on how to optimize the performance [1,2].
- The vPars monitor does not do any distribution of processors and memory based on locality. It requires the system administrator to use the CLM and CLP options to appropriately configure partitions. Hence, when adding or removing CLM from a cell, take into consideration the CLP

as well. Assigning processors from one cell and cell local memory from another cell might lead to less optimal performance due to increased distance between processor and memory locality. The HP-UX kernel optimizes and performs better when it has processors, memory, and if possible I/O from the same locality. For more information on processor configuration in a vPars environment refer to the CPU Configuration Guidelines for vPars white paper [3].

## Base and Floating Memory

Starting with the A.05.xx release, online migration of memory is supported on vPars. This means memory can be added to and deleted from a live vPars partition without requiring reboot. There are two constraints. First, the HP-UX must be memory migration capable to make use of this feature. Memory migration capability is present in releases starting with HP-UX 11i v3. Second, on HP Integrity servers, firmware must also be memory migration capable. Please refer to the *HP-UX Virtual Partitions Ordering and Configuration Guide* [4] for the firmware version required for memory migration.

When memory is added, the HP-UX kernel discovers the new memory, updates its data structures to include the new memory pages, and allows the applications to make use of the new memory that became available. When memory is to be deleted, the HP-UX kernel selects the memory pages to evacuate, moves the contents in that memory to some other free pages or to disk, and then removes these memory pages from its data structures. After the kernel completes the memory delete operation, the memory is marked as free and can be assigned to other partitions. Executing the `vparstatus` command with the `-A` option after the completion of the delete operation displays the memory available for assignment.

During deletion, there are certain memory contents that cannot be evacuated to another free page. Examples of such contents include kernel code and certain kernel data structures. While allocating memory during boot or during run-time, the HP-UX kernel needs to know in advance what memory to use for such contents that cannot be evacuated. To aid the HP-UX kernel in this differentiation, the vPars software subdivides ILM and CLM into two types: base memory and floating memory.

The system administrator while picking memory for the partition now has four choices: (i) the amount of ILM that will be base memory, (ii) the amount of ILM that will be floating memory, (iii) the amount of memory from each cell that will be base memory and (iv) the amount of memory from each cell that will be floating memory. The ILM and CLM options in the vPars commands have a new attribute that allows the system administrator to specify whether the memory being added or removed is base or floating. The amount of memory that is specified as base and floating is stored in the vPars database. Hence, the amount of memory the partition gets as base and floating persists across partition and monitor reboots.

During boot or when the memory is added online, the monitor notifies the kernel which memory pages are base and floating. The kernel uses base memory pages for contents that it cannot evacuate. On the other hand, for contents that it can evacuate, the kernel will use either floating or base memory. Hence, memory that is marked as base cannot be deleted when the partition is live. It can only be deleted when the partition is down. Memory that is marked as floating can be deleted when the partition is live or when the partition is down.

There is one key differentiation in management of ILM and CLM versus base and floating memory in the system. If the system administrator has to change the amount of ILM and CLM in the system, it will require a reboot of the system and the monitor<sup>2</sup>. For example, if the system contains 4 GB of ILM which is divided among partitions and the system administrator wants to increase the ILM to 6 GB and reduce the amount of CLM in the system by 2 GB, it will require a system and monitor reboot. On

---

<sup>2</sup> Reboot is necessary because firmware must re-interleave the memory and can do so only with a hard partition reset.

the other hand, changing the amount of memory that is base and floating does not require a system or monitor reboot. Any available memory in the vPars monitor can be used as base or floating while assigning it to a partition. When the partition boots or when the memory is added online, the kernel gets the specified amount as base or floating. Once the memory is removed from the partition and becomes available, it can be used either as base or floating while assigning to other partitions.

The following lists some of the base/floating memory rules:

1. When no attribute is specified the memory defaults to base. This provides backward compatibility for earlier vPars releases where all memory was treated as base by the kernel and the kernel did not allow removal of memory when live. Hence, base memory can be added or deleted without specifying any attribute or by explicitly including the `':b[ase]'` attribute. The following lists syntaxes to add base ILM and CLM.

```
# vparcreate -p <vPar> -a mem::

```

2. Floating memory requires explicit specification of the attribute `':f[loat]'` during add/delete. The following lists syntaxes to add floating ILM and CLM.

```
# vparcreate -p <vPar> -a mem::

```

3. Base memory can be added when the partition is live or when it is down. To delete base memory, the partition must be down.
4. Floating memory can be added or deleted when the partition is live or down.
5. Base and floating memory can be added or deleted in one command line.

```
# vparmodify -p <vPar> -a mem::

```

6. The HP-UX kernel accepts either an add or a delete operation but does not accept both simultaneously. Hence, add and delete cannot be performed in the same command when the partition is live. For example, if the partition is live, the add and delete operations must be separated into two commands as given below.

```
# vparmodify -p <vPar> -a mem::

```

Combining the above operations into one single command as given below fails.

```
# vparmodify -p <vPar> -a mem::

```

In order to illustrate base and floating memory, let us revisit the configuration discussed in the previous section and set up a sample vPars system with base and floating memory. To reiterate the setup, the system contains 2 GB of CLM from cell 0, 2 GB of CLM from cell 1, and 4 GB of total ILM. Now, in order to create the two partitions (vpar1 with 512 MB of base and 1 GB of floating ILM and vpar2 with 1 GB of base and 512 MB of floating ILM) we would invoke the following `vparcreate` commands:

```
# vparcreate -p vpar1 -a mem::512 -a mem::1024:f
# vparcreate -p vpar2 -a mem::1024 -a mem::512:f
```



To add 1 GB as base and 512 MB as floating CLM from cell 0 to vpar1 and 512 MB as base and 512 MB as floating CLM from cell 1 to vpar2, we would do the following:

```
# vparmodify -p vpar1 -a cell:0:mem:::1024 -a cell:0:mem:::512:f
# vparmodify -p vpar2 -a cell:1:mem:::512 -a cell:1:mem:::512:f
```

Assuming both vpar1 and vpar2 are live, to remove 512 MB of floating ILM from vpar2 and add it to vpar1 as base ILM, we would do the following:

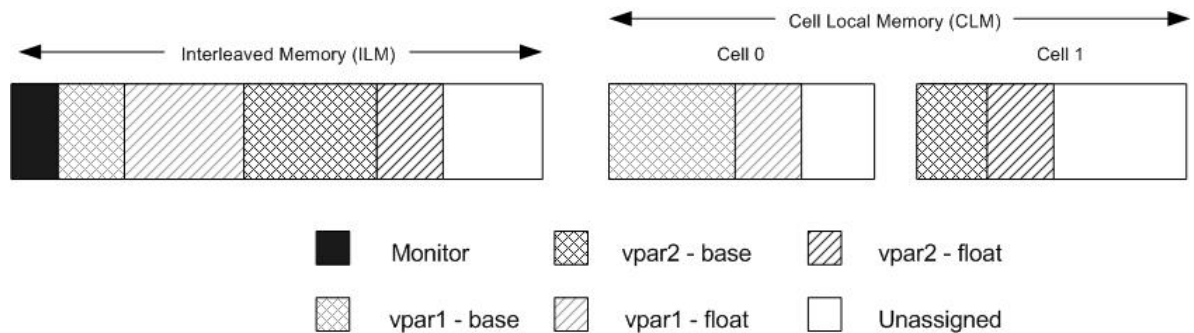
```
# vparmodify -p vpar2 -d mem:::512:f
# vparmodify -p vpar1 -a mem:::512
```

To remove 512 MB of base ILM from vpar1 and add it to vpar2 as floating ILM, we would shut down vpar1 and once it is down issue the following commands to transfer memory from vpar1 to vpar2 and then boot vpar1 back up:

```
# vparmodify -p vpar1 -d mem:::512
# vparmodify -p vpar2 -a mem:::512:f
```

Figure 2 below is an extension of Figure 1 and shows the memory distribution for the above example with base and floating ILM and CLM when both vpar1 and vpar2 are live. At the end of the last operation, vpar1 contains 512 MB base ILM, 1024 MB floating ILM, 1024 MB base CLM from cell 0 and 512 MB floating CLM from cell 0. vpar2 contains 1024 MB base ILM, 512 MB floating ILM, 512 MB base CLM from cell 1 and 512 MB floating CLM from cell 1. As will be explained later, the vPar monitor tries to allocate base memory below the floating memory within each locality when the partition is booted.

Figure 2 Base and Floating Memory Partitioning



### Base and Floating Memory Guidelines

The following provides some guidelines on how to use base and floating memory in a vPars system:

- The system administrator is encouraged to configure enough base memory during partition boot to achieve the required baseline application performance taking into consideration the following two constraints:
  - ∅ The kernel has more flexibility in using base memory. The kernel restricts the use of floating memory to contents that it can later relocate if it is selected for deletion. Hence,



a system with all base memory might perform better compared with another system with the same amount of memory but divided between base and floating memory.

- ∅ Some kernel sub-systems and applications do their allocations based on memory discovered during boot time. These subsystems or applications might allocate their cache based on the amount of base memory available to the kernel during boot time and might not scale that cache when more base memory is later added online. Hence, the performance of a system that is booted with less memory followed by online add of memory might not be equal to a system with the same amount of memory but all available during boot.
- For processor and I/O resources in a given locality, the kernel might optimize the memory access latency by allocating data structures from base memory within that locality. Hence, it is better to configure some amount of base memory from a cell if the partition is going to have processors and I/O from the same cell.
- Initial measurement has shown that the HP-UX kernel takes about 1 second to add or delete 1 GB of memory on an idle or lightly loaded system. In addition to the time taken by the HP-UX kernel, the vPars commands and monitor might take additional time of up to a second to process addition or deletion. The times quoted here can increase in future releases. On the other hand, on a system with heavy memory pressure, the HP-UX kernel might take minutes or even hours to evacuate memory. Hence, it is advisable not to delete memory on a loaded system. Such a constraint does not exist while adding more memory. The system administrator can add memory anticipating memory load or during memory load.
  - ∅ When system takes more time to delete or does not make forward progress on deletion, it may be helpful to cancel the operation (cancel is discussed in the “Memory Migration Management” chapter) and split the single large memory delete operation into two or more small memory delete operations. For example, if the partition contains a large amount of floating memory, instead of deleting all the floating memory in one delete operation, it may help to split it into multiple small delete operations each consisting of 10% to 25% of floating memory.
- The HP-UX kernel requires a certain percentage of total memory to be base memory to assure expected system performance and ensure that there is adequate memory for critical system needs. The following table shows the minimum amount of memory that must be configured as base memory for a given total memory size. Some applications may need more base memory than what is recommended below.

Physical Memory (total)	Base Memory (minimum)
1.5 GB to 3 GB	1.5 GB <sup>3</sup>
3 GB to 8 GB	½ of physical memory
8 GB to 16 GB	4 GB
>16 GB	¼ of physical memory

During boot, if enough memory is not configured as base, the kernel might convert some of the floating memory into base. When that happens, the vPars monitor updates the vPars database to reflect the increase in base memory and decrease in floating memory for that partition.

<sup>3</sup> The HP-UX 11i v3 requires minimum of 1.5GB of base memory.

## Memory Granules – Containers for Physical Memory

The granule (aka segment) denotes the unit of memory by which the user can assign or remove memory resources to a partition. In the A.01.xx, A.02.xx and A.03.xx vPars releases, the vPars software fixes the granule size at 64 MB and does not provide any flexibility to the system administrator to change it. The vPars A.04.xx and A.05.xx releases allow the system administrator to specify the granule size as a power of two multiple of 64 MB: 64 MB, 128 MB, 256 MB, and so on. Moreover, the system administrator can choose different granule sizes for ILM and CLM.

The vPars software logically partitions the physical memory address space in the system into equal sized and aligned granules and uses these as containers for valid physical memory within that address space. For example, if the granule size is 64 MB, the physical address space is divided into 0 to 64 MB for the first granule, 64 MB to 128 MB for the second granule and so on. If during boot, the vPars monitor discovers valid physical memory from 32 MB to 128 MB, then the first granule will contain 32MB of memory and second granule will contain 64MB of memory. When a granule is not fully populated to its size with memory as demonstrated with the first granule in the above example, the granule is said to contain a memory hole.

The ILM and CLM granule size information specified by the system administrator is stored in the vPars database when it is created. For A.04.xx and A.05.xx vPars releases, the default size is 128 MB for both ILM and CLM. On HP Integrity servers, the firmware also divides the memory into memory objects based on the granule size. Hence, to inform firmware, in addition to the vPars database, the granule size information is also stored in non-volatile RAM (NVRAM). The following provides details on how to configure granule size in the A.04.xx and A.05.xx vPars releases. As mentioned earlier, this configuration is not required in the A.01.xx, A.02.xx and A.03.xx vPars releases as the granule size is fixed at 64 MB.

1. On HP Integrity servers, there is a limit in firmware on the number of memory objects that can be created. Determine what the limit is as follows:
  - a. Find out the maximum CLM any cell has and the total ILM in the system using the `parstatus` command.

```
# parstatus -c <cell number>... -V
# parstatus -p <hard partition number> -V
```
  - b. Change the mode to vPars and issue the `vparsenv` command to find out the maximum number of ILM and per cell CLM granules that can be created on the system.

```
# vparenv -m vPars
# vparenv
```
  - c. Divide the total ILM by the maximum number of ILM granules to determine the minimum ILM granule size.
  - d. Divide the maximum CLM found in any cell by the maximum number of per cell CLM granules to determine the minimum CLM granule size.
2. Determine the ILM and CLM granule size for the system making sure it is a power of two multiple of 64 MB. For HP Integrity servers, it should be equal to or greater than the minimum size evaluated in the previous step.

3. Create the first partition and the vPars database and specify the ILM and CLM granule size using the `-g` option. On HP Integrity servers, specify the `y` attribute to update the firmware NVRAM with the granule size.

```
# vparcreate -p <vPar> -D <database> -g ilm:<size>[:y] -g clm:<size>[:y] ..
```

4. When ready to move to a new granule size, shut down the system and boot the monitor with the new database.

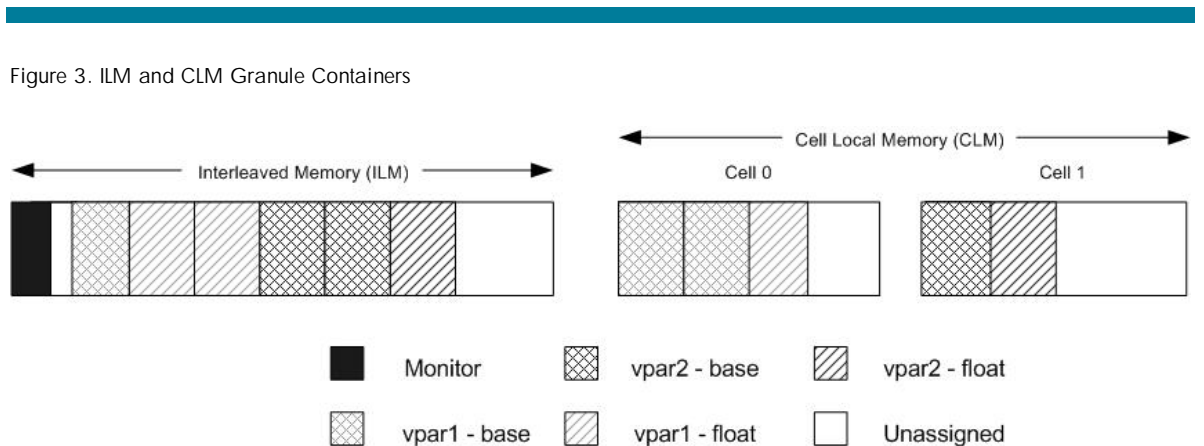
To illustrate further, let us consider the same setup discussed in previous chapters. Since the setup is an HP Integrity server, we need to find the minimum granule size using the `vparsenv` command. We already know that the maximum CLM in any cell is 2 GB and total ILM is 4 GB on the example system.

```
# vpsenv
vpsenv: The next boot mode setting is "vPars".
...
vpsenv: Note: The maximum possible CLM granules per cell is 256.
vpsenv: Note: The maximum possible ILM granules for this system is 1024.
```

The example system is already set to vPars mode for next boot. The maximum number of CLM granules per cell is 256 and maximum number of ILM granules is 1024. Hence, dividing the size by maximum granules, we get 8 MB for CLM (2 GB divided by 256) and 4 MB (4 GB divided by 1024) for ILM. This is well below the supported minimum of 64 MB. Assuming that we want to create a new database called `vpdb.512` with both ILM and CLM granule size of 512 MB and update the NVRAM as well:

```
# vparcreate -p vpar1 -D /stand/vpdb.512 -g ilm:512:y -g clm:512:y ...
```

Figure 3 below is an extension of Figure 2 and shows the memory distribution with 512 MB CLM and ILM granules as logical containers of underlying physical memory. As can be seen, the monitor and firmware occupies 384 MB in the first granule. Hence, the first granule is said to contain a memory hole of 384 MB with 128 MB as assignable memory. That granule is not yet assigned to any partition.



## Memory Granules Guidelines

The following provides some guidelines on setting the granule size:

- On HP Integrity servers, if the system contains a significant number of granules, it might increase the boot time of partitions that are running the prior HP-UX 11i v2 release. Hence, if the partition is running HP-UX 11i v2 and boot speed is a strong requirement, choose a large granule size. The significant number of granules does not impact the boot time of partitions running HP-UX 11i v3 and might not impact the boot time of partitions that run future HP-UX 11i v2 releases.
- A very large granule size limits the number of partitions that can be created or might impact the amount of memory a partition gets. For example, if the system contains 4 GB of ILM, choosing 1024 MB as the ILM granule size limits the number of partitions to 4 or less since the system will contain four granules each with memory equal to or less than 1024 MB. Moreover, memory contained in one or more granules can be less than 1024 MB because of memory consumed by vPars monitor and firmware. Hence, do not set the granule size such that all partitions are limited to a couple of granules at best. If not much memory movement will occur between partitions once they are created, set the granule size such that each partition gets at least a few granules.
- If periodic movements of memory between partitions (live or down) is a strong requirement, set the granule size to the minimum amount of memory that will be moved between partitions. For example, if your configuration is a small memory system and 128 MB is the minimum amount of ILM that will be moved between partitions, then set the ILM granule size to 128 MB. On the other hand, if it is a large memory system and 2 GB is the minimum amount of ILM that will be moved between partitions, then set the ILM granule size to 2 GB.
- Changing granule size requires system and monitor reboot. If you plan to use different granule sizes during different periods of time, create a vPars database for each granule size and boot the vPars monitor with that database when the switch is required. On HP Integrity servers, booting the monitor with a different granule size requires an update of granule size in NVRAM. Unless that is done, the vPars monitor will reboot the system when it finds that the granule size in NVRAM differs from the granule size in the vPars database. The `vparsenv` command can be used to change the granule size in the NVRAM, as shown below, before rebooting the system to use the new database containing the new granule size:
 

```
# vpsenv -g ilm:<size> -g clm:<size>
```
- The HP-UX 11i v3 kernel limits the large page size to the size of granule. Hence, when the granule size is small it can sometimes lead to loss in performance especially if the partition contains applications that benefit from large page sizes.
- On PA-RISC platforms, each partition's HP-UX kernel requires memory below 2 GB to boot and run. Memory below 2 GB is ILM. A portion of the first memory granule will be taken by the vPars monitor for its code and data and hence cannot be used for the partition's HP-UX kernel. Excluding the first ILM granule, there should be at least one ILM granule below 2 GB for each partition's kernel. For example, if the granule size is 256 MB, there are 8 granules below 2 GB. Excluding the first granule, there are seven granules. On such a system, the system administrator can create a maximum of seven partitions assuming the size of each partition's kernel is less than 256 MB. Hence, while choosing ILM granule size on PA-RISC platforms, the system administrator should also consider the maximum number of vPars partitions planned for the system<sup>4</sup>.

---

<sup>4</sup> The size of the granule does not impact boot time of HP-UX 11i v2 kernel on PA-RISC platforms.

# Memory Resource Assignment

In a vPars system, by default, a partition does not get any memory when it is created. Hence, the memory required for a partition must be explicitly added. There are two parts to the memory assignment: the amount and the actual address ranges that form this amount. This chapter describes how to assign memory while creating the database, how the vPars monitor divides the memory that it finds among partitions during monitor boot, how memory ranges get bound or unbound during partition boot, shut down, online add and delete, and some of the side effects of the binding.

The first section describes the division of memory between partitions while creating and modifying a non-live<sup>5</sup> database. The second section describes the memory operations on a live database including partition boot, online add, and online delete.

## Memory Allocation – Non-live Database

The vPars software does not validate the resources that are assigned to each partition against the resources that are present in the system where the non-live database is being modified. With respect to memory, the system administrator must take inventory of the physical memory of each type (ILM and CLM in each cell) on the target system and reduce from that the memory that will be taken by firmware and vPars monitor prior to creating the database and assigning memory to the partition. The memory assigned to each partition is known as requested memory and vPars software allows the modification without checking whether it can allocate the requested memory when this database is actually used later to partition the target system.

When the vPars monitor is booted, it takes inventory of the memory of each type available for use. The monitor then allocates each partition's request based on the order in which the partitions are found in the database. The base memory request of each partition in the database is satisfied before any floating memory request. If, during allocation, the amount of memory available is less than the amount of memory requested whatever available is allocated to the partition. Hence, if total ILM or CLM memory requested in the database is more than the available ILM or CLM in the system, one or more partitions at the end gets less memory or zero memory.

In the example setup in previous chapter, the system contains: 4 GB ILM, 2 GB of CLM from cell 0 and 2 GB of CLM from cell 1. Assuming the system administrator creates vpar1 before vpar2 and then assigns 1 GB of ILM as base and 1GB of ILM as floating to vpar1 and vpar2. After the monitor boot, the available ILM is 3712 MB because 384 MB out of the 4 GB is used by the monitor and firmware. The vPars monitor allocates base memory first. Hence, vpar1 and vpar2 each get 1 GB of base memory. However, for floating memory, vpar1 will get 1 GB of floating memory whereas vpar2 will get 640 MB. Hence, vpar2 gets less floating memory than what it requested. To avoid such oversubscription, the system administrator can do one of the following:

1. Do not assign all the memory in the system among partitions. Instead leave some amount of memory free for use by vPars monitor and firmware (described in more detail later in Memory Accounting Chapter) while assigning memory to partitions.
2. Create just one partition and assign the required memory to the partition. After that, boot the monitor and the partition and take inventory of the actual available memory using `vparstatus -A` and then create rest of the partitions assigning to them available memory.

---

<sup>5</sup> A database is non-live or alternate database under following two conditions: (i) HP-UX is booted stand-alone (not under vPars monitor) and vPars commands are used to create and modify the database. (ii) HP-UX is booted under vPars monitor, however, the database being created or modified is the database that the vPars monitor is not currently using.

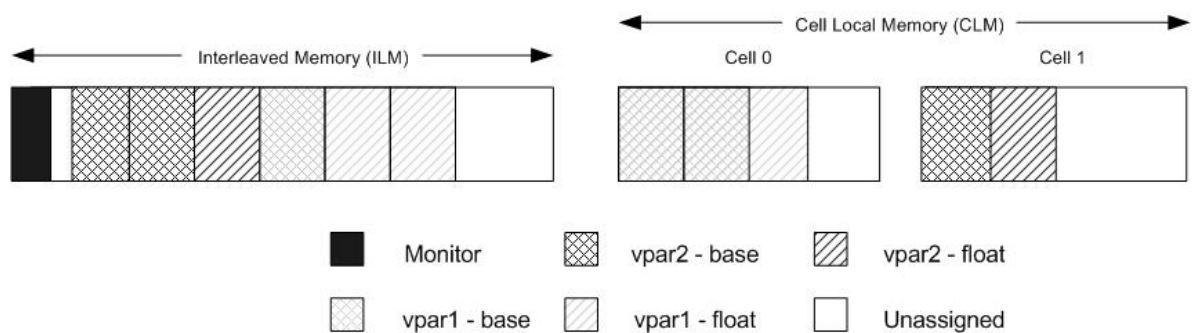
## Memory Allocation and Binding – Live Database

Once booted, the vPars monitor maintains a memory copy of the file database. Any resource modification to the partition in this live database is validated against the available resources. Hence, when ILM or CLM memory is added, the vPars monitor checks whether the requested amount is available. If not, the system administrator knows when the vPars command returns if the memory allocated is less than the requested amount.

When a partition is down, the allocated memory indicates the amount, type, and locality of memory that has been reserved for it. It does not indicate the actual address ranges that will form this allocated memory. The memory granules that make up the address ranges are bound to the partition when the partition is booted. Hence, when the partition is down, the `vparstatus -v` output only indicates the amount of memory reserved to the partition. When the partition is live, the output (only in A.04.xx and A.05.xx vPars releases) contains the actual address ranges that form the amount of memory. To bind granules, the vPars monitor starts from the low address end of each ILM and CLM memory type and binds the available granules for base memory first and then binds the granules for floating memory. The vPars monitor binds the base memory at the low end because during early boot firmware and HP-UX kernel allocates memory at the low end for contents that they cannot evacuate. When the partition shuts down, the binding between the granules and the partition is removed. The next time the partition boots, it is not guaranteed that the partition will get same granules and hence the address ranges.

For example, as shown in Figure 3 in Memory Granules Chapter, when vpar1 is booted first, the monitor starts from low end of ILM and CLM on cell 0 and binds the first available granules to base and then binds the floating granules. When partition vpar2 is booted next, monitor binds the ILM granules above vpar1 ILM granules as base followed by floating. Figure 4 below shows what happens when both vpar1 and vpar2 are shutdown and then vpar2 is booted first and then vpar1. As can be seen the granules and hence the address ranges vpar1 and vpar2 get are different because the binding is removed when partitions go down and a rebinding is done based on the available granules at the time the partition is booted.

Figure 4. Change of Address Ranges after Reboot



When a partition is live, the allocation is immediately followed by binding during memory add. Similarly when memory is deleted, the unbinding of the granules is followed by a reduction in amount of memory that the partition owns. During add, the vPars monitor starts from the low address and binds the available granules as each command option is processed. Then the HP-UX kernel in the partition adds these memory ranges to its usable physical memory. During deletion, the vPars monitor passes the amount of memory specified to the HP-UX kernel and the HP-UX kernel selects the granules

to delete to optimize the deletion time. Hence, the granules selected for deletion need not be in any specific order.

The following lists some of the side effects related to memory granule binding:

1. As evident from the example, due to the dynamism of memory binding, the ILM and CLM address ranges associated with a partition change across reboots, deletes and adds. Hence, the system administrator should not rely on a partition getting the same memory address ranges.
2. If the system contains one or more memory granules with holes, the sum of the address ranges during add, delete, or boot can vary slightly than the request. Suppose the granule size is 512 MB and the system contains four available granules: 512 MB, 480 MB, 400 MB and 420 MB. The following details these variations:
  - a. If the partition that is being booted is allocated 1024 MB, it might get 512 MB and 480 MB granules, the sum of which is 992 MB instead of the 1024 MB.
  - b. During the next boot, if the only available granules are 400 MB and 420 MB, then that partition will get a total of 820 MB instead of the 992 MB it got during previous boot.
  - c. If the add request to a live partition is 512 MB and the available granules are 400 MB and 420 MB, the partition might get the 420 MB granule.
  - d. If the delete request is 512 MB and the floating granule sizes in the live partition are 512 MB and 400 MB, the kernel might select 400 MB granule to delete if it can evacuate it faster.
3. As explained in the previous chapter, on PA-RISC platforms, the memory granules below 2 GB are used only for the partition's kernel. Hence, during boot, the vPars monitor binds only the required number of granules below 2 GB to load the partition's kernel and the remaining are picked from memory granules above 2 GB.



## User Specified Ranges

The previous chapter described how the system administrator can specify the amount of memory, the type and the locality required for the partition and let the vPars monitor choose the ranges that will be part of the partition. Instead of letting the vPars monitor pick the ranges, the system administrator can explicitly specify one or more address ranges, known as user specified ranges, within which all or a portion of the requested memory should reside. This chapter describes when to use them, how to use them and the side effects of using them.

The preferred way to bind memory to the partition is to let the vPars monitor choose the memory range during boot and online add instead of explicitly binding it using user specified ranges as described in this chapter. Explicit binding should only be done if there is a specific circumstance that necessitates it. Following is one such example:

On PA-RISC platforms, the vPars monitor assigns one or more granules below 2 GB based on the current size of the kernel. If later the kernel size<sup>6</sup> is increased significantly and rebooted, the vPars monitor might not be able to find enough contiguous granules below 2 GB to fit the new kernel if all the memory in the system is being used by active partitions. To avoid this problem, the user can explicitly bind contiguous memory below 2 GB taking into account future kernel growth of the partition. Note that each partition will require memory below 2 GB to boot. Hence, binding more memory below 2 GB to one partition might make other partitions unable to boot.

The following provides the various syntaxes to add or delete memory ranges. Each range consists of the starting address of the range and the size of the range and optional attribute to specify whether it is base or floating. The default is base for add. For delete, the type is optional because the vPars software can determine the type of the range based on the address range being deleted.

```
# vparcreate -p <vPar> .. -a mem::<>address>:<size>[:b|base|f|float] ..
# vparmodify -p <vPar> .. -a|-d mem::<>address>:<size>[:b|base|f|float] ..
```

For example, to create a partition with 1 GB of base ILM memory, 1 GB of floating ILM memory, and then assign user specified base range of 512 MB at starting address 0x20000000 and user specified floating range of 512 MB starting at starting address 0x100000000, the commands would be:

```
# vparcreate -p vpar1 .. -a mem::1024 -a mem::1024:f ..
# vparmodify -p vpar1 .. -a mem::0x20000000:512 -a mem::0x100000000:512:f ..
```

The behavior of the user specified range depends on whether it is a non-live database or a down partition or a live partition. The following lists some of the user specified range rules:

1. The size of the range must be a multiple of the granule size. Both end points of a range must be aligned on a granule boundary.
2. When a partition is down or belongs to a non-live database, addition or deletion of a range does not increase or decrease the amount of memory the partition owns. It only results in early binding of the address range.
  - a. In the example above, if vpar1 is down, after the addition of a range, the total ILM owned by the partition continues to be 1 GB base and 1 GB floating. When the partition is booted, instead of binding granules for the whole 2 GB, the vPars monitor will only bind granules for the remaining 512 MB base and 512 MB floating because

---

<sup>6</sup> Refer to the HP-UX Virtual Partitions Administrator's Guide [\[4\]](#) Appendix C for details on how to determine kernel size.

the user has already explicitly bound 512 MB of address range as base and 512 MB of address range as floating.

3. When the partition is live, add or delete of a user specified range results in an increase or decrease of memory that the partition owns. In the example above, if vpar1 is live, at the end of the operation vpar1 will own 1.5 GB of base and 1.5 GB of floating ILM.
4. When a partition is down, the partition should contain enough unbound memory of that type (base or floating ILM or CLM) before a range can be added. In the above example, assuming vpar1 is in down state, the range add will fail, if the size of the base or floating range exceeds 1 GB as that is more than the 1 GB of base or floating allocated to the partition. If another command is issued in the above example to add a range, the size of the base or floating range cannot exceed 512 MB because that is the amount of base or floating ILM that is not yet bound.
5. When a partition is down or belongs to a non-live database, the allocated memory of a given type cannot go below the sum of the address ranges. In the above example, a command to delete memory as given below fails because deleting 768 MB of floating ILM memory takes the allocated floating ILM to 256 MB which is less than the 512 MB address range bound to the partition.

```
# vparmodify -p vpar1 -d mem::768:f
```

6. When a partition belongs to a live database, there should be valid memory within the specified range. A range cannot overlap a memory hole.
7. User specified ranges should be deleted only by using the delete range option. There is an exception to this rule:

During boot, the HP-UX kernel might convert any floating granule to base granule (as described in Memory Granules chapter). Similarly during delete from a live partition, the HP-UX kernel might select any floating granule to optimize the deletion time. If the floating granule that was selected by the HP-UX kernel belongs to a user specified floating range, the vPars monitor deletes the user specified floating range and the explicit binding between the granules and the partition. The granules will still be part of the partition as monitor-bound granules.

8. The vPars software does not validate the range for valid memory or type of memory while adding to a non-live database. Hence, the system administrator should ensure that the target system contains valid memory of that type within the specified end points. During monitor boot, the requested range is deleted if there is no valid memory within the range or if the addition of the range results in the amount of bound base or floating ILM or CLM to exceed the amount of base or floating ILM or CLM that has been allocated. In the above example, if there is no valid memory at address 0x20000000, the first range is deleted. If the memory at address 0x100000000 belongs to cell 0, the second range is deleted because no CLM from cell 0 is allocated to the partition.

# Memory Migration Management

This chapter describes the tools available to the system administrator to manage and monitor memory migration before, during, and after the operation. Following are some of the tools available to the system administrator to manage and monitor memory migration:

- `vparstatus` command output (`-v` and `-A` options).
- GlancePlus performance monitor (`gpm -rpt MemoryReport`).
- EVM(5) event management progress logs.
- Cancel operation in the `vparmodify` command (`-C` option).

The following sections describe in detail when and how these tools can be used. The last section in this chapter demonstrates the usage of these tools through an example memory migration operation.

## Tools to Determine Memory Usage

Before adding or removing memory from the partition, it is critical for the system administrator to determine the memory usage in the partition. The memory usage on a live partition can be determined using GlancePlus reporting tool. The GlancePlus has two memory related reports – `MemoryReport` and `MemoryUsageGraph`. The example detailed in the last section of this chapter uses `MemoryReport`. The `MemoryReport` lists the amount of memory free (`Free Mem`), amount of memory used by the kernel (`Sys Mem`) and user (`User Mem`) and amount of memory used by buffer and file cache (`Buf Cache` and `FileCache`). The report also indicates whether pages are getting swapped to disk (`Paged Out`) which is the case when system is low on free memory. `MemoryUsageGraph` depicts similar information in graphical form.

Before issuing the command to delete memory from live partition:

1. Find the amount of floating memory the partition has in each locality using `vparstatus` (`-v` or `-M` option). The amount selected from each locality to delete should be less than or equal to this amount.
2. Find the amount of free memory (`Free Mem`) in the partition using GlancePlus (`gpm -rpt MemoryReport`). If the amount being deleted is less than free memory, skip the rest of the steps below.
3. Find the current file cache size (`FileCache`) using GlancePlus (`gpm -rpt MemoryReport`).
4. Find the minimum file cache size (`kctune filecache_min filecache_max`).
5. Add the free memory from step 2 to the current file cache size from step 3 and subtract the minimum file cache size from step 4 to determine the amount of freeable memory.
6. If the memory being deleted is more than the amount calculated in step 5, reduce the amount to delete to less than or equal to that amount.

Before issuing the command to add memory to live partition:

1. Find out the amount of free memory in the partition using GlancePlus (`gpm -rpt MemoryReport`) and then decide on the amount to add.

2. Find out the available amount of memory in each locality using the `vparstatus` (`-A` option) output. The amount selected from each locality to add should be less than or equal to this available amount.

## Monitoring the Status of an Online Operation

The `vparstatus` command output (`-v` option) shows the status of the last initiated CPU or memory migration operation. In the `vparstatus` output, this shows up under the section called "OL\* Details". A sample output is shown below:

```
[OL* Details]
Sequence ID: 123

Operation: Memory Addition

Status: PASS
```

In the above output, the sequence ID is a number used to uniquely identify the memory migration (or CPU migration) operation that was last initiated on a given partition. The operation field indicates what type of online migration event was initiated, and the status field indicates the current state of the operation. The status field may contain PASS, PENDING, ABORT or FAIL. The pending status indicates that the operation is in progress. If the status shows ABORT or FAIL, this means the online memory (or CPU) migration has failed. When this happens, there are three main logs that the system administrator may consult to determine what exactly prevented the operation from completing successfully: the EVM event log (`evmget` | `evmshow`), the operating system log (`/var/adm/syslog/syslog.log`) and the vPars monitor log (`vparextract -l`).

## Cancelling an Online Operation

Memory migration, and in particular the memory delete operation, could sometimes take a long time to complete. This is because for the memory delete operation, the kernel has to first evacuate the contents of the memory pages that are in use by applications. Depending on the size of the memory being deleted, its usage and the amount of memory that is free, this operation could end up taking considerable time. The `vparmodify` command contains an option to allow the user to cancel an ongoing CPU or memory migration. The option (`-C sequenceid`) will allow the user to cancel individual pending CPU or memory OL\* operations on a per partition basis using the sequence identifier for the operation. The operation can be cancelled under two conditions: the operation is still in the PENDING state and the operating system has not committed to the operation. The status of the operation can be determined using output of the `vparstatus -v` command. The progress of the operation can be determined using the output of the `evmget` | `evmshow` command or `evmget -A` command from within the partition being modified.

## Memory Migration Example

This section illustrates the usage of commands and tools through a memory migration example. The memory migration operation is demonstrated as follows:

1. Describe the experimental setup.
2. Describe memory usage on vpar1 that has 1 GB of free memory.
3. Describe memory usage on vpar2 that uses all memory.
4. Describe memory usage on vpar1 after online deletion of 1 GB of memory.
5. Describe memory usage on vpar2 after online addition of 1 GB of memory.

At each step, appropriate commands are executed to look at the memory usage and monitor the progress of the operation. Only the relevant output from the command is shown.

### Experimental Setup

The setup used for this experiment is a system with 12 GB ILM and three partitions: vpar1 with 2 GB of base memory and 1 GB of floating memory, vpar2 with 2 GB of base memory and 1 GB of floating memory, and vpar3 with 3 GB of base memory and remaining memory assigned as floating memory. The following shows the output of the `vparstatus` command with the memory distribution.

```
vpar1# vparstatus
[Virtual Partition]
```

Virtual Partition Name	Memory (MB)			
	ILM		CLM	
	# User Ranges/MB	Total MB	# User Ranges/MB	Total MB
vpar1	0/ 0	3072	0/ 0	0
vpar2	0/ 0	3072	0/ 0	0
vpar3	0/ 0	5490	0/ 0	0

### Memory Usage on vpar1

The `vparstatus` output below shows the portion of the total memory that is floating memory on vpar1. The GlancePlus Memory Report Graph confirms that a little over 1.1 GB of memory is not used by the partition and that the kernel is not paging the memory to disk (Paged out is 0).

```
vpar1# vparstatus -v -p vpar1
[Memory Details]
.....
ILM Total (MB): 3072 (Floating 1024)
.....

vpar1# gpm -rpt MemoryReport &
```

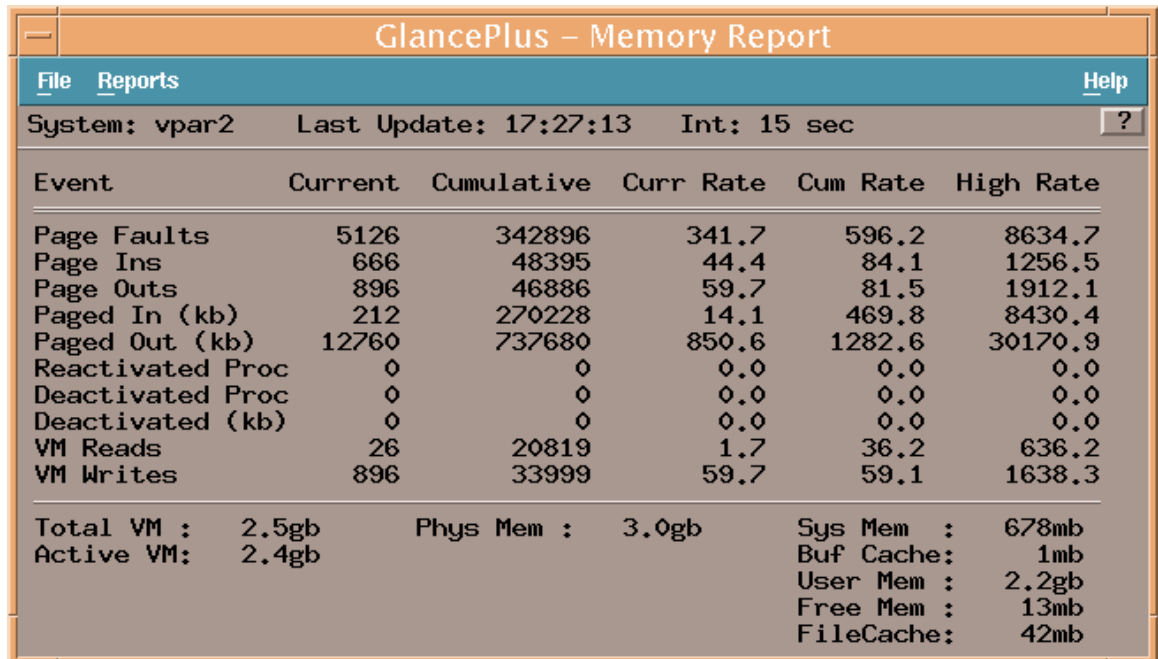
GlancePlus – Memory Report						
File Reports						Help
System: vpar1		Last Update: 17:25:57		Int: 15 sec		?
Event	Current	Cumulative	Curr Rate	Cum Rate	High Rate	
Page Faults	32	78831	2.1	217.7	4581.3	
Page Ins	15	7241	1.0	19.9	1336.0	
Page Outs	0	0	0.0	0.0	0.0	
Paged In (kb)	240	7000	16.1	19.3	1573.3	
Paged Out (kb)	0	0	0.0	0.0	0.0	
Reactivated Proc	0	0	0.0	0.0	0.0	
Deactivated Proc	0	0	0.0	0.0	0.0	
Deactivated (kb)	0	0	0.0	0.0	0.0	
VM Reads	34	1045	2.2	2.8	31.2	
VM Writes	0	0	0.0	0.0	0.0	
Total VM :	1.3gb	Phys Mem :	3.0gb	Sys Mem :	900mb	
Active VM:	1.1gb			BuF Cache:	1mb	
				User Mem :	957mb	
				Free Mem :	1.1gb	
				FileCache:	58mb	

## Memory Usage on vpar2

The `vparstatus` output below shows the portion of the total memory that is floating memory on vpar2. The GlancePlus Memory Report confirms that partition has just 13 MB of free memory and that the kernel is writing contents of the pages to disk (`paged out` is 12760 KB) to free up memory for applications.

```
vpar2# vparstatus -v -p vpar2

[Memory Details]
.....
ILM Total (MB): 3072 (Floating 1024)
.....
vpar2# gpm -rpt MemoryReport &
```



Event	Current	Cumulative	Curr Rate	Cum Rate	High Rate
Page Faults	5126	342896	341.7	596.2	8634.7
Page Ins	666	48395	44.4	84.1	1256.5
Page Outs	896	46886	59.7	81.5	1912.1
Paged In (kb)	212	270228	14.1	469.8	8430.4
Paged Out (kb)	12760	737680	850.6	1282.6	30170.9
Reactivated Proc	0	0	0.0	0.0	0.0
Deactivated Proc	0	0	0.0	0.0	0.0
Deactivated (kb)	0	0	0.0	0.0	0.0
VM Reads	26	20819	1.7	36.2	636.2
VM Writes	896	33999	59.7	59.1	1638.3

Total VM :	2.5gb	Phys Mem :	3.0gb	Sys Mem :	678mb
Active VM:	2.4gb			Buf Cache:	1mb
				User Mem :	2.2gb
				Free Mem :	13mb
				FileCache:	42mb

## Deletion of Memory from vpar1

1 GB of floating memory is deleted from vpar1 using `vparmodify`. The output of `vparstatus` shows the PENDING state followed by the PASS state of the memory deletion operation with sequence identifier 1. Once completed, the output of `vparstatus` shows that the partition has 2 GB of base memory and no floating memory. The GlancePlus Memory Report below confirms the reduction in free memory to 178 MB as a result of the memory online delete operation. The remaining memory is enough to satisfy the current load and the system has not reached the stage requiring paging (`paged out` is still zero). The EVM logs show the detailed progress of the memory online delete operation to delete 1 GB of memory starting at the physical address 0x150000000. Finally, the `vparstatus` output shows that the monitor now has 1 GB of memory available that can be added as base or floating memory to other partitions.

```
vpar1# vparmodify -p vpar1 -d mem::1024:f

vpar1# vparstatus -v -p vpar1
[Memory Details]
.....
ILM Total (MB): 3072 (Floating 1024) (migration pending)
.....
[OL* Details]
Sequence ID: 1
```

Operation: Memory Deletion

Status: PENDING

```
vpar1# vparstatus -v -p vpar1
[Memory Details]
.....
ILM Total (MB): 2048 (Floating 0)
.....
[OL* Details]
Sequence ID: 1
```

Operation: Memory Deletion

Status: PASS

```
vpar1# gpm -rpt MemoryReport &
```

GlancePlus - Memory Report					
File		Reports		Help	
System: vpar1		Last Update: 17:29:29		Int: 16 sec	
Event	Current	Cumulative	Curr Rate	Cum Rate	High Rate
Page Faults	3778	90409	236.1	157.6	4581.3
Page Ins	773	9779	48.3	17.0	1336.0
Page Outs	0	0	0.0	0.0	0.0
Paged In (kb)	3924	12220	245.2	21.3	1573.3
Paged Out (kb)	0	0	0.0	0.0	0.0
Reactivated Proc	0	0	0.0	0.0	0.0
Deactivated Proc	0	0	0.0	0.0	0.0
Deactivated (kb)	0	0	0.0	0.0	0.0
VM Reads	553	1788	34.5	3.1	34.5
VM Writes	0	0	0.0	0.0	0.0
Total VM :	1.3gb	Phys Mem :	2.0gb	Sys Mem :	850mb
Active VM:	1.2gb			Buf Cache:	1mb
				User Mem :	913mb
				Free Mem :	178mb
				FileCache:	58mb

```
vpar1# evmget | evmshow
```

OLAD: The olad infrastructure is locked and ready to accept parameters for the operation. No other olad operations may be initiated on this nPartition until the operation is complete. The sequence number for this operation is 1.

OLAD: The olad infrastructure has set the parameters for the operation. The sequence number is 1 and the parameters are "vpar memory delete operation".

OLAD: The olad infrastructure has started the requested operation with sequence number 1.

.....  
OLAD: The olad infrastructure is performing the execution phase in the vm subsystem. Execution of 0 of 262144 Memory Pages to Delete are complete.

.....  
OLAD: The olad infrastructure is performing the execution phase in the vm subsystem. Execution of 262144 of 262144 Memory Pages to Delete are complete.  
OLAD: The olad infrastructure is performing the final actions to complete the operation. It is no longer possible to cancel this operation.

OLAD: Memory 170000:10000 has gone offline

OLAD: Memory 160000:10000 has gone offline

OLAD: Memory 150000:10000 has gone offline

OLAD: Memory 180000:10000 has gone offline

.....



```
vpar1# vparstatus -A
.....
[Available ILM (Base /Range)]: 0x150000000/1024
                               (bytes) (MB)
[Available ILM (MB)]: 1024
.....
```

### Addition of the Freed Memory to vpar2

1 GB of available memory is added as floating memory to vpar2 using `vparmodify`. The output of `vparstatus` shows the PASS state of the memory addition operation with sequence identifier 1. As described earlier, the sequence id is unique within each partition and not across partitions. Hence, both vpar1 and vpar2 in this example happen to have the same sequence id number. Once completed, the output of `vparstatus` shows that the partition has 2 GB of base memory and 2 GB of floating memory. The GlancePlus Memory Report below confirms that the free memory has increased to 921 MB and the paging to swap space has stopped after the online addition of memory. The EVM logs show the detailed progress of the memory online addition operation to add 1 GB of memory starting at the physical address 0x150000000.

```
vpar2# vparmodify -p vpar2 -a mem::1024:f
```

```
vpar2# vparstatus -v -p vpar2
[Memory Details]
.....
ILM Total (MB): 4096 (Floating 2048)
.....
[OL* Details]
Sequence ID: 1
```

Operation: Memory Addition

Status: PASS

```
vpar2# gpm -rpt MemoryReport &
```

GlancePlus – Memory Report					
File Reports					Help
System: vpar2		Last Update: 17:30:59		Int: 15 sec	
Event	Current	Cumulative	Curr Rate	Cum Rate	High Rate
Page Faults	2553	358123	171.3	447.3	8634.7
Page Ins	1132	54454	75.9	68.0	1256.5
Page Outs	0	47051	0.0	58.7	1912.1
Paged In (kb)	616	292148	41.3	364.9	8430.4
Paged Out (kb)	0	739732	0.0	923.9	30170.9
Reactivated Proc	0	0	0.0	0.0	0.0
Deactivated Proc	0	0	0.0	0.0	0.0
Deactivated (kb)	0	0	0.0	0.0	0.0
VM Reads	69	23476	4.6	29.3	640.4
VM Writes	0	34164	0.0	42.6	1649.3
Total VM :	2.5gb	Phys Mem :	4.0gb	Sys Mem :	733mb
Active VM:	2.4gb			Buf Cache:	1mb
				User Mem :	2.3gb
				Free Mem :	921mb
				FileCache:	44mb

```
vpar2# evmget | evmshow
```

OLAD: The olad infrastructure is locked and ready to accept parameters for the operation. No other olad operations may be initiated on this nPartition until the operation is complete. The sequence number for this operation is 1.  
OLAD: The olad infrastructure has set the parameters for the operation. The sequence number is 1 and the parameters are "vpar memory add operation".  
OLAD: The olad infrastructure has started the requested operation with sequence number 1.  
.....  
OLAD: Memory 180000:10000 has come online  
OLAD: Memory 170000:10000 has come online  
OLAD: Memory 160000:10000 has come online  
OLAD: Memory 150000:10000 has come online  
.....

## Memory Accounting

On any given system not all physical memory is available for application use. In a non-vPars system, firmware takes some memory for its code and data structures before it hands over remaining memory to the OS kernel and the OS kernel uses some memory for its code and data structures. The memory taken by the kernel depends on the amount of memory and other resources (processors and memory) in the system. In a vPars system, the vPars monitor that resides between operating system and firmware consumes some amount of memory. The memory consumed by the vPars monitor can vary from one vPars release to another. Currently, this ranges from 18MB to 40MB on a PA-RISC platform and from 384MB to 512MB on an HP Integrity server. Each partition in a vPars system has its own instance of OS kernel (that is, kernel code and data is not shared among partitions). Hence, the kernel consumes each partition's memory for its code and data. In addition, on an HP Integrity server, each partition has its own instance of firmware which consumes around 32 to 64 MB. Again, this can vary among firmware releases and platforms.

This chapter describes the tools available in A.04.xx and A.05.xx vPars releases to find the actual memory assigned to each partition and the amount of memory consumed by the vPars monitor and by the firmware. The tools described here do not account for the memory taken by each HP-UX OS instance during boot or the memory consumed by the firmware instance of each partition. The following lists the steps to account memory in a vPars system:

1. Find the total amount of ILM in the system using `parstatus -p <partition number> -v`.
2. Find the CLM from each cell in the system using `parstatus -c <cell number> -v`. Add the CLM of each cell to get the total CLM.
3. Add the ILM and the total CLM to get the total memory in the system.
4. Find the memory bound to all the partitions using `vparstatus -v` command. Add the size of user specified as well as monitor assigned memory ranges of all partitions to get the total memory that has already been bound to the partitions.
5. Find the memory not bound to any partition using `vparstatus -A` command. Add the size of each range to get the total memory not bound to any partition.
6. Find the memory consumed by monitor and firmware using `vparstatus -m` command. Add the size of each range which is in bytes and then convert to megabytes (MB).
7. Add the amount of memory bound to each partition (step 4), the amount of memory not yet bound to any partition (step 5), and the amount of memory consumed by firmware and vPars monitor (Step 6) to get the total memory seen by vPars monitor during boot.
8. The memory computed in step 7 should be equal to or very close to total memory in the system computed in step 3.

To illustrate further, let us consider the similar setup discussed in the previous sections that has 4 GB of ILM and 2 GB of CLM from cell 0 and 2 GB CLM from cell 2. The granule size is 512MB. The total memory in the system is 8 GB (8192 MB). The partition vpar1 contains 1 GB of ILM and 1 GB of CLM from cell 0 and partition vpar2 contains 1GB of ILM and 1 GB of CLM from cell 2. Running the command `vparstatus-v` to find out the memory ranges bound to the partition results in the following output (only memory related information is shown here).

```
Name:          vpar1
State:         Up
```

```

[Memory Details]
ILM, user-assigned [Base /Range]: 0x40000000/512
                                   (bytes) (MB)
ILM, monitor-assigned [Base /Range]: 0x20000000/512
                                   (bytes) (MB)
ILM Total (MB): 1024 (Floating 0)
CLM, monitor-assigned [CellID Base /Range]: 0 0x7008000000/1024
                                   (bytes) (MB)
CLM (CellID MB): 0 1024 (Floating 0)

```

```

Name:          vpar2
State:         Down

```

```

[Memory Details]
ILM, user-assigned [Base /Range]: 0x4080000000/512
                                   (bytes) (MB)
ILM, monitor-assigned [Base /Range]:
                                   (bytes) (MB)
ILM Total (MB): 1024 (Floating 0)

CLM, user-assigned [CellID Base /Range]: 2 0x7408000000/512 (Floating)
                                   (bytes) (MB)
CLM, monitor-assigned [CellID Base /Range]:
                                   (bytes) (MB)
CLM (CellID MB): 2 1024 (Floating 1024)

```

For vpar1, which is live (UP state), the memory range 0x40000000/512 has been explicitly bound by the user and the memory ranges 0x20000000/512 and 0x7008000000/1024 have been bound by the monitor. For vpar2, which is down, memory ranges 0x4080000000/512 and 0x7408000000/512 have been explicitly bound by the user. Hence, adding up the size of all bound memory ranges, the total memory bound to partition is 3072 MB. Running the command `vparstatus -A` to find the memory ranges not yet bound results in the following output.

```

[Available ILM (Base /Range)]: 0x60000000/511
                                   (bytes) (MB)          0x40a0000000/1536
[Available ILM (MB)]: 1535

[Available CLM (CellID Base /Range)]: 0 0x700c000000/512
                                   (bytes) (MB)          0 0x700e000000/440
                                   2 0x740a000000/1024
                                   2 0x740e000000/448

[Available CLM (CellID MB)]: 0 952
                                   2 960

```

Adding up the size of all the ILM and CLM memory ranges, the total memory not yet bound to any partition is 4471 MB. Running the command `vparstatus -m` to find the memory ranges taken by vPar monitor and firmware results in the following output.

```

Memory ranges used: 0x0/349519872 monitor
                   0x14d54000/434176 firmware
                   0x14dbe000/532480 monitor
                   0x14e40000/253952 firmware
                   0x14e7e000/1425408 monitor
                   0x14fda000/50487296 firmware
                   0x18000000/134213632 monitor
                   0x7fffe000/8192 firmware
                   0x700fc00000/67108864 firmware
                   0x740fc00000/67108864 firmware

```

Adding up the size in bytes of all the memory ranges and converting it into megabytes results in 640 MB. Adding up the memory bound to partition, memory not yet bound to any partition, and memory used by monitor and firmware yields 8183 MB, which is close to the 8192 MB physical memory in the system. Hence, not all memory installed in the system can be accounted using vPars command output as some memory used by hardware or firmware is not reported in the output. For this setup the

difference is 9 MB. This difference can be more on some systems or platforms. The following lists some of the reasons why the difference can be more:

- When the amount of memory each cell contributes to interleaving is not uniform, some amount of memory is lost during interleaving. For example, on a 3 cell system where one cell contributes 0.5 GB, a second cell contributes 1 GB, and the third cell contributes 2 GB, some amount of memory will not be available after interleaving. The exact amount that will not be available depends on how non-uniform the contribution from each cell is.
- The memory reported by `vparstatus -v` and `vparstatus -A` is aligned on the megabyte boundary. Hence, if there are memory holes of few kilobytes, they are reported as a megabyte hole in the output. This might exacerbate the difference during memory accounting if there are a significant number of double bit error memory pages on HP Integrity servers, as such pages are shown as memory holes. However, the actual usable memory, except for the memory holes of a few kilobytes, is still available and used by each OS instance in the partition.

## References

1. See Chapter 12 in HP-UX 11i Version 2 September 2004 Release Notes: HP 9000 Servers, HP Integrity Servers, and HP Workstations located at <http://docs.hp.com/en/5990-8153/index.html>.
2. See [http://docs.hp.com/en/4913/ccNUMA\\_White\\_Paper.pdf](http://docs.hp.com/en/4913/ccNUMA_White_Paper.pdf) for the white paper titled "ccNUMA Overview".
3. See [http://docs.hp.com/en/8767/cpu\\_config.pdf](http://docs.hp.com/en/8767/cpu_config.pdf) for the white paper titled "CPU Configuration Guidelines for vPars".
4. See <http://docs.hp.com/en/hpux11iv3.html#Virtual%20Partitions> for vPars administration documents and white papers specific to the HP-UX 11i v3 release stream.

© 2007 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Linux is a U.S. registered trademark of Linus Torvalds. Microsoft and Windows are U.S. registered trademarks of Microsoft Corporation. UNIX is a registered trademark of The Open Group.

xx9832-xx003, March 2007

