Extracting Resource Allocation Data for Chargeback in a HP Virtual Server Environment for HP Integrity Servers

Abstract.............................................................................................................................................. 3

Chargeback and VSE Concepts.............................................................................................................. 3
  Chargeback .................................................................................................................................... 3
  Components of a complete chargeback solution........................................................................... 3
  Chargeback and budget forecasting............................................................................................... 4
  Chargeback policy impact on data collection needs ........................................................................ 4

Chargeback with Virtualized Platforms of HP VSE ................................................................................ 9
  The asset being shared .................................................................................................................... 9
  User workloads ............................................................................................................................. 9
  Automated control and resource guarantees .................................................................................. 11
  Choosing which resource parameter to measure ............................................................................ 11

VSE Data for Chargeback Reports.................................................................................................. 12
  Reporting Style 1: IT as Service Provider ....................................................................................... 12
  Reporting Style 2: Cost Recovery .................................................................................................. 13

Summary of Tools for Extracting Resource Allocation Data .................................................................. 15
  Capacity Advisor ....................................................................................................................... 16
  Global Workload Manager (gWLM) ............................................................................................ 16
  HP Integrity Virtual Machines (Integrity VM) .............................................................................. 16
  HP Systems Insight Manager (HP SIM) ........................................................................................ 16
  Virtualization Manager ............................................................................................................... 16
  Instant Capacity (iCAP) ............................................................................................................... 16

Using VSE Tools to Gather Chargeback Information............................................................................... 17
  Configuration Assumptions ............................................................................................................ 17
  Listing Workloads ....................................................................................................................... 18
    GUI: VSE Manager Workload tab ............................................................................................. 18
    CLI: gwlmreport config command ............................................................................................ 19
  Workload Utilization Information .................................................................................................. 21
    GUI: Capacity Advisor Profile Viewer ................................................................................... 21
    CLI: capreport tool .................................................................................................................. 23
  Workload Allocation Information .................................................................................................. 26
    GUI: Capacity Advisor Utilization Report ............................................................................... 26
    CLI: capreport tool .................................................................................................................. 26
  Workload Resource Guarantees with gWLM .................................................................................. 28
Abstract

Chargeback is of growing interest to IT organizations that are implementing shared services on virtualized servers. A key element of a chargeback solution is the collection of usage data when workloads belonging to multiple different end users share the same physical resources. This paper describes the data of interest and the tools used to report it in a computing environment that implements the HP Virtual Server Environment (VSE) suite of platform virtualization technologies.

Chargeback and VSE Concepts

Virtualization and chargeback represent two of the most significant changes being integrated into IT organizations today. Virtualization allows IT organizations to increase the utilization of their server platform resources and to become more flexible in responding to business needs. At the same time, chargeback becomes more important and more complex as the traditional “one server, one end-user organization” relationship is less and less the norm.

Chargeback

Chargeback is the practice of charging individual business units for the IT services they use. The definition of chargeback is simple, but the IT services being measured and associated chargeback solution can be complex. This section discusses the position of VSE in an overall chargeback solution.

Components of a complete chargeback solution

To fully measure the usage of business units and to bill them, a chargeback solution requires multiple subcomponents, including a database of resources, asset management and accounting information, user organizations, a rate and cost structure, a billing/invoicing engine, and a data collection mechanism. Figure 1 illustrates these components and their relationship. Of these components, this paper describes only the data that VSE can provide to a data collection mechanism.

![Figure 1 Chargeback components](image-url)
An additional area of complexity is the IT service itself. To deploy email, a data warehouse, order entry, and any of the myriad typical IT services requires many components and layers that all have hardware, software, human capital, services, support, and datacenter costs. Figure 2 depicts the many types of resources which can contribute to the cost of IT providing a service to a business unit. The server platform VSE layer discussed in this paper is just one of many about which IT might gather information for chargeback.

![Figure 2 Service implementation components](image)

**Chargeback and budget forecasting**

Some IT organizations gather data about users and resources but use it for future budgeting rather than for charging for past usage. In these cases, IT generally looks at data over the past fiscal year in order to provide user organizations with a figure that is cross-charged to them by IT. The data collected is largely the same as the case where IT wants regularly charge for past usage, and the tools and techniques described in this paper can be used for this budget forecasting purpose as well.

**Chargeback policy impact on data collection needs**

Along with the complexity of the IT service and chargeback solution, there are also many policy decisions to be made. These policy decisions determine which VSE parameters are necessary. Although it is outside the scope of this paper to describe all possible policy decisions in chargeback, the following three considerations help explain the parameter choices under discussion.

**Server Inventory Information**

The first consideration is whether information about the physical server inventory is necessary. An end bill to a business unit does not necessarily require a list of the specific physical assets that were used to provide the service for which they are being charged. On the other hand, IT might find this data necessary if they choose to allocate the costs of all physical assets to the pool of users. In addition, IT might this information beneficial from the standpoint of accounting, budgeting, and future
optimization of resources. Figure 3 illustrates the two different possible styles of reporting. The internal IT report includes a summary of which resources a service used. The report for the business unit that is being charged might not require this information.

![Diagram of resource mapping](image)

**Figure 3 Mapping use to specific assets**

An example involving virtualization technology is shown in Figure 4. The example involves a server with 6 CPU cores and 3 workloads. Two different approaches to reporting and billing are possible. One reporting option is to concentrate on recovering to IT the cost of the server. Assuming that CPU core usage is used to determine workload usage, the total core count of the server must be gathered and the usage of each workload divided by this amount to identify a percentage value. An alternative reporting option is to list workloads and their usage and to apply a charging rate independent of the particular server and its total pool of resources. Assuming again that CPU core usage is being used to determine chargeback, all that is required for each workload is the average number of cores it used during the reporting period.
Usage Parameters: Utilization, Allocation, and Guarantees

The second policy consideration that impacts the choice of parameters is the form of the resource usage parameter. Of the three parameters described here, some are used more often than, but under certain circumstances any of them might be chosen by IT.

- Utilization usage: This is the actual measured usage of resources by a user. Utilization is the most intuitive and potentially most commonly used form of usage. Utilization is also the most useful form for future optimization of resource allocation and planning.

- Allocated usage: For some server and virtualization technologies, resources can be allocated to users in greater increments than they actually consume. As an implementation detail of IT and “overhead,” avoid passing these allocations on to the end-user consumer. On the other hand, IT might need to assign all costs of assets they deploy to the pool of users who use them. In this case, allocated usage can represent a more direct metric of the costs of resource deployment.

- Guaranteed usage: This is the amount of resources guaranteed to be available to the user independent of how much the user actually consumes. When a guarantee exists, it can represent the value that determines the sizing of the infrastructure and represents the critical metric of what the service is costing IT to implement.

To illustrate these different forms of usage and why IT might choose one over another in different situations, consider the following analogy to a member of a tennis club illustrated in Figure 5. If the member is guaranteed a certain number of hours of court time per month, the club is likely to be very sensitive to the number of members and associated guarantees relative to the number of courts and the total schedulable court time. This factor is important for determining charges. If the courts are allocated in hour-long blocks and the member does not use an entire hour of reserved time, the club
cannot effectively use the remainder of the time for any other purpose. The user might prefer to be billed for utilized court time in increments of minutes, but the club allocates time in hourly increments.

Monthly member charge based on;
- Minutes on a court (utilized)?
- Minutes allocated (courts blocked in hour increments)?
- Minimum hours of access guaranteed in membership?
- Combination?

Figure 5 Tennis club analogy

The next example is specific to servers managed with VSE technology and is depicted in Figure 6. Using the Virtual Partition (vPar) technology, the CPU cores of a server can be allocated dynamically to different vPars running different operating systems (OSs) and associated applications. The cores are allocated in whole-core increments. The OS and applications in a vPar utilize a percentage of the cores of the vPar. This utilization value is typically less than 100% of the core allocated to the vPar. Utilization is the amount actually used by the user, but the core allocation is the resource provided by IT. As with unused minutes of reserved time on a tennis court not being usable by any other members, the unused processing of vPar cores is not usable by any other vPar. In addition, vPars can be combined with VSE Global Workload Manager (gWLM) technology. When used with gWLM, the vPar can have a guaranteed number of cores (referred to as “owned”) that exceeds both utilization and allocation. The number of guaranteed cores requires additional capacity sizing by IT independent of usage of the vPar. This sizing relates directly to IT’s cost of providing the vPar to the user.
Which usage parameter should IT use?
• Utilization – actually used by workload
• Allocation – what IT actually had to provide
• Guarantee – what IT might have had to size the box for…

![Graph showing three lines labeled Utilization, Allocation, and Guarantee over time](image)

**Figure 6 Choices when choosing a usage parameter with vpars**

These two examples are not meant to imply that one of these parameters is always the correct one to use. Rather, they illustrate that the choice of parameter varies from situation to situation. To highlight this further, consider a final example shown in Figure 7 in which IT might not choose to use any usage data at all. An IT department is deploying large numbers of very uniform servers with multiple virtual machines. The virtual machines are the service for which business units are charged. The virtual machines might be so commoditized and the underlying server asset value might be so small that IT chooses to simply charge a flat fee per virtual machine, independent of usage. In this case, the data of interest from VSE is primarily an inventory of workloads.

Charge based on which of the following?
• VM usage of underlying server host (CPU, memory etc.)
• flat fee per VM

![Diagram of hosts and VMs](image)

**Figure 7 Charging based on usage or a flat fee**

Background:
• Large numbers of midrange or blade systems with multiple VMs
• Each vm is service to be charged to a business unit
• Servers, VMs and applications are very standardized in the organization
Data-Gathering and Reporting Intervals

The third policy consideration is the billing interval. A chargeback accounting period is dependent on both internal IT and the accounting practices of a company. Gathering of resource-allocation data can happen at any interval equal to or shorter than the billing period. This paper assumes a data-gathering period of one month or less.

Chargeback with Virtualized Platforms of HP VSE

HP VSE is a suite of technologies for creating highly efficient and agile platform infrastructures with a central theme of configuring and managing multiple workloads on common hardware resources. For more information about the VSE suite, see the white paper called, “An Introduction to the HP Virtual Server Environment,” available on the HP Documentation website:

http://docs.hp.com/en/11011/IntroToVSE.pdf

The asset being shared

From an accounting perspective, the hardware resource that VSE technology allows users to share is the physical server. VSE physical servers currently belong to one of two architecture families: HP Integrity servers or HP 9000 servers. Within both architecture families, larger physical servers can be partitioned into smaller servers by using the nPar and vPar technologies. These systems are referred to as cell-based complexes. For purposes of this paper, the end asset from a cost-accounting perspective to IT organizations is the complex, not individual subpartitions.

The potential secondary resource being shared is Temporary Instant Capacity (TiCAP). TiCAP is a finite processing capability that is consumed as cores without usage rights are activated and deactivated. From an asset perspective, this processing can be purchased and tracked separately from the physical server.

User workloads

An IT service might have multiple components in different layers. At the VSE level, the unit of work that can be assigned to an end-user organization is called a workload. VSE workloads are of the following types:

- Server: The whole OS workload of a nonpartitioned server
- nPar: The whole OS workload of an nPar
- vPar: The whole OS workload of a vPar
- VM guest: The processes of an HP Integrity virtual machine as seen on the Integrity VM Host
- pset: The processes of a processor set
- fss: The processes of a Fair Share Scheduler group

Whole OS workloads consist of an entire OS instance and the applications that are installed and are running on it. The pset and fss workloads are processes of an application instance within an OS. A VM guest is a virtual machine. The virtual machine as seen from the underlying VM Host is a process. As seen from the VM guest itself, the virtual machine is an entire OS instance and the applications that are installed and running on it.
### Workload Hierarchies

One way that the VSE architecture helps determine resource usage for specific users is by way of hierarchical nesting. Figure 8 shows the possible combinations in the VSE hierarchy from physical server through different workload types.

<table>
<thead>
<tr>
<th>Complex Server</th>
<th>nPar</th>
<th>FSS</th>
<th>Integrity VM</th>
<th>vPar</th>
<th>vPar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Server</td>
<td>Server</td>
<td>Complex</td>
<td>Server</td>
<td>Complex</td>
</tr>
</tbody>
</table>

* HP Integrity VM is HP Integrity Server Only

#### Figure 8 Possible workload hierarchies in VSE

Because of the hierarchies, VSE can contain intermediate workloads that are not associated with billable end users from the IT perspective. Later sections of the paper will discuss how to identify or filter out these workloads from the output of the VSE tools. An addition kind of workload which is not associated with billable end users is the “Other” workload type. VSE can create “Other” workloads for its own management purposes when a whole OS workload is subdivided into virtual machines, psets, or fss workloads. VSE uses these Other workloads to contain everything in the whole OS workload that is not associated with a user-defined workload. Other workloads are not typically of interest for chargeback purposes.

Figure 9 is an example of an environment with two physical servers, one a complex and one not which illustrates that not every workload in a VSE environment is of interest from a billing prospective.
Figure 9 Example hierarchies and workloads of interest

The noncomplex server is subdivided using Integrity VM with three VM guests. VSE associates four workloads with this server: one workload for the underlying whole OS workload, one each for the three VM guests, and one Other workload. Of these four workloads, only the three VM guests represent user workloads.

The complex server is subdivided into two nPartitions. One of the nPartitions is a whole OS workload running fss workloads. The other is subdivided into two vPars, each of which is a whole OS workload. Of all the workloads that VSE associates with the configuration, only the two fss workloads and the two vPar workloads are of interest from a billing perspective.

Automated control and resource guarantees

A powerful feature of VSE is its automated control. Automated control allows IT to set up resource pools that can be shared between multiple workloads. The primary automated control engine of VSE is the gWLM component. This technology can control all workload types.

An additional component of VSE that can result in guarantees is Integrity VM and its entitlement capability. This technology only applies to VM guest workloads which are not already under gWLM control. If the VM guest workload is under gWLM control, the gWLM guarantees take precedence and those of Integrity VM’s entitlement capability are ignored.

Choosing which resource parameter to measure

The resource being shared is the physical server that can have many performance utilization attributes to use in determining each user’s usage. The existence of multiple parameters does not mean that all are needed or useful, so you can use discretion in selecting them. Figure 10 uses the analogy of a car rental, which applies a combination of rental period, mileage, and fuel level and omits parameters such as tire wear, oil level, and transmission fluid level. In both the car rental and server chargeback situations, multiple usage parameters are available, but not all are necessarily needed.
Many usage parameters are available for both cars and servers, but for charging only a minimum set are typically needed to determine relative usage of the asset.

Many usage parameters are available for both cars and servers, but for charging only a minimum set are typically needed to determine relative usage of the asset.

In most environments, a suitable primary indicator of a workload’s usage of a physical server is CPU count. Server cost is generally proportional to CPU count and is sized according to the anticipated CPU requirements of the workloads that will use it. The remainder of this paper refers to CPU in units of CPU cores and calls them cores.

A possible secondary indicator of a server’s workload usage is memory utilization. For some workloads, memory utilization is either a better indicator or is useful as a modifier for determining a workload’s use of a physical server and its associated cost.

Network and storage usage parameters of a workload relate more directly to WAN and SAN asset costs – fibre cables, switches, arrays, LUNs, and so on – and therefore are of lesser importance when assigning cost of the physical server itself. With that in mind, this paper focuses solely on the physical server and does not address accounting for workload network and SAN utilization. For those interested specifically in chargeback in SANs, HP StorageWorks Essentials includes a Chargeback Manager component for charging back SAN-based storage.

VSE Data for Chargeback Reports

The exact parameters of interest to an IT organization – the form of internal IT chargeback reports and user-facing chargeback bills – vary based on the services, policies, and VSE technologies being used. This section describes two possible reporting styles.

**Reporting Style 1: IT as Service Provider**

This style of report lists all workloads and their level of usage independent of any particular server. Workload usage is reported in cores or memory, or both, as absolute values; for example, 3.2 cores and 1.5 GB used. In its simplest form, this information is multiplied by a rate to result in the charge amount for the owning organization. The resulting amount can be used for any configuration but is most likely used in environments where:
Large-scale deployment of workloads and servers means that the cost is generally known without data from a specific hosting server. Server hardware costs are small relative to other deployment costs. Multiple instances of similar or commodity workloads are deployed.

**Reporting Style 2: Cost Recovery**

This style of report is concerned with which server a workload ran on so that the charge to the owning organization can be based on the cost of the specific server. In this case, the data used to compile the report must include the configuration of the server in CPU cores or memory so that the workloads usage can be converted into a percentage. In addition, the report must specify the server’s serial number or asset tag information so that its cost can be retrieved from an IT asset management system. This style of report can be used in any configuration but is most likely used in environments where:

- Deployments of high-end servers mean that the cost of the server is a significant percentage of the total cost of hosting the workload.
- Varied workload and server configurations require that data specific to each configuration be gathered to understand relationship of usage to cost.
- The report style is dictated by the accounting practices.

The following table lists types of data that can be gathered from VSE and their applicability to the two different reporting approaches described here:
<table>
<thead>
<tr>
<th>VSE Data</th>
<th>Service Provider Style Report</th>
<th>Cost Recovery Style Report</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>service model where charge is determined by level of service provided</td>
<td>user charges based on proportional use of specific server</td>
</tr>
<tr>
<td>List of Workloads</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Workload Usage:</td>
<td>Utilization</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>readily available for all workloads, represents level of service used by consumer</td>
<td>readily available for all workloads - sum of all for a physical server would typically be less than 100% but can be scaled to achieve 100% recovery</td>
</tr>
<tr>
<td>Allocation*</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>neither actually used by user (see utilization above) or a implicit guarantee to user (see guarantee below)</td>
<td>server resources allocated to a workload can not be assigned to any other</td>
</tr>
<tr>
<td>Workload Guarantees**</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td>could be of interest based on level of service guaranteed and therefore a premium charge over actual utilization</td>
<td>physical servers sized for guaranteed levels of service</td>
</tr>
<tr>
<td>List of Physical Servers</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Mapping of Workloads to Physical Servers</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>Physical Server Attributes</td>
<td>Cores and Memory</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>charge independent of specific hardware</td>
<td>need server configuration to compute proportional use by individual workloads</td>
</tr>
<tr>
<td></td>
<td>Serial Number and Asset Tag</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td>charge independent of specific hardware</td>
<td>need to identify server in asset management and account systems</td>
</tr>
<tr>
<td>iCAP ***</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>charge independent of specific hardware</td>
<td>need to account for resources with versus resources without usage rights</td>
</tr>
<tr>
<td>TiCAP ****</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td></td>
<td>charge independent of specific hardware</td>
<td>need to identify consumption of specially purchased computing capacity by workloads</td>
</tr>
</tbody>
</table>

* nPar and vPar workloads only
** gWLM managed or HPVM entitlement workloads only
*** cell-based servers with iCAP only
**** cell-based servers with TiCAP only
Summary of Tools for Extracting Resource Allocation Data

VSE includes many configuration and management tools to gather the data described in the previous table. The following table describes the tools and components discussed in this paper. All of the GUI tools referenced in the preceding table are either part of the HP SIM GUI interface or integrated add-ons. With the exception of the `hpvmstatus` and `icapstatus` command-line tools, all the CLI commands referenced run on the HP SIM CMS host command line. In the case of `hpvmstatus` and `icapstatus`, HP SIM’s Custom Tool and Task mechanisms can be used to launch these tools on designated VSE systems from the HP SIM CMS.

<table>
<thead>
<tr>
<th>VSE Component</th>
<th>Description</th>
<th>GUI Access (HP SIM)</th>
<th>CLI Access</th>
</tr>
</thead>
</table>

| **Capacity Advisor** | Records usage information for workloads. The tools provided with Capacity Advisor allow the recorded usage information to be displayed and reports generated. | Optimize  
Capacity Advisor  
View Profile  
Create Utilization Reports | capprofile, capreport tools |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Global Workload Manager (gWLM)</strong></td>
<td>Maintains a list of the workloads in the VSE environment and their basic properties. For workloads under its automated control gWLM can provide guaranteed resource allocations (Owned values) and usage information.</td>
<td>Optimize gWLM</td>
<td>gwlreport config, gwlmd discover tools</td>
</tr>
<tr>
<td><strong>HP Integrity Virtual Machines (Integrity VM)</strong></td>
<td>For VM guest workloads, enforces entitlement values which represent guaranteed resource allocations.</td>
<td>In Integrity VM Manager, click VM on the virtual machine name in Virtualization Manager.</td>
<td>hpvmstatus tools</td>
</tr>
<tr>
<td><strong>HP Systems Insight Manager (HP SIM)</strong></td>
<td>Maintains basic system identification and description information as well as optionally entered asset management data. HP SIM also maintains association information useful determining the hierarchical relationship of workloads to physical servers.</td>
<td>All Servers System(s)</td>
<td>mxnode, mxquery, mxreport, mxtask tools</td>
</tr>
<tr>
<td><strong>Virtualization Manager</strong></td>
<td>Graphically depicts a VSE environment with hierarchical relationship of workloads to physical servers. The interface also provides entry points for HP SIM and other VSE tools.</td>
<td>Tools VSE Management</td>
<td>-</td>
</tr>
<tr>
<td><strong>Instant Capacity (iCAP)</strong></td>
<td>Manages licensing of CPU cores on physical servers through the concept of usage rights. Logs record the number of cores and amount of memory with usage rights in a complex. In addition, iCAP monitors and reports TiCAP capacity configuration and consumption.</td>
<td>-</td>
<td>icapstatus, /var/adm/icap.log tools</td>
</tr>
</tbody>
</table>
Using VSE Tools to Gather Chargeback Information

The remainder of this paper describes in detail how to generate the data needed for a VSE report into a chargeback system. The GUI and CLI tools are discussed in each case. Using CLI commands allows users to write scripts that automate extraction of the information into a consolidated report. A useful, related automation framework is the HP SIM Custom Tools and Tasks.

Configuration Assumptions

In this section, the following assumptions are made about the VSE environment:

- The version of the HP VSE Management Software is 3.0.
- All managed nodes of interest have been discovered and identified within HP SIM and have VSE agents loaded and configured.
- Any or all of the following technologies might be in use on the managed nodes: nPars, vPars, VM guests, psets, fss, iCAP, and TiCAP.
- gWLM is installed and licensed even if no SRDs have been configured.
- Capacity Advisor is installed and licensed, and data collection is enabled on all managed nodes of interest.
- With the exception of HP SIM CLI commands, which do not require root privileges, all others are run as the local root account on the CMS. All GUI commands are run with an HP SIM user with all HP SIM privileges.
Listing Workloads

The units of work defined and managed by VSE are workloads. Getting a complete list of all the workloads in the VSE environment is the first step in reporting usage information.

In addition, the workloads of interest must be identified in the list of all workloads. This is because not all VSE workloads listed are necessarily directly related to users. This is for two main reasons: some workloads are intermediate in a hierarchy, and some are Other workloads that are part of the VSE implementation and therefore are automatically created.

GUI: VSE Manager Workload tab

The Workload tab in VSE Virtualization Manager, shown in the following figure, displays a table with all the workloads in the environment of all types.

In this output, the Other workloads all end with .OTHER in their name and can be ignored. This output can be used with the Contained In column to learn more about the hierarchies of the workloads, but an easier approach is to display the hierarchy on the System page, as shown in the following figure:
In this output, the leaf workloads of the environment are easily identified. The Other workloads (va01sv.zko.hp.com.OTHER, va02cm.zko.hp.com.OTHER) and intermediate workloads (va00-cplx, va01sv, va02cm) in the hierarchy typically are not associated with an end user.

**CLI: gwlmreport config command**

The output from the *gwlmreport* tool with the `config` flag contains a subsection listing workloads in the environment. For example:

```
CMS # gwlmreport config
Generating historical config report.
Please be patient. This may take several minutes.

Current Configuration
=======================

### Config Date: 2007/09/21 16:06:39 EDT
```
Of the workloads listed, it is necessary to identify the workloads that are of interest as end-user workloads, and not as VSE-created Other workloads or intermediate workloads.

A simple filter can first organize the output of the `gwlmreport config` command into one line for each workload with workload name, type, and host at the same time removing workloads with .OTHER in the name. For example:

```bash
CMS # cat listwklds1.sh
```

```
gwlmreport config | awk '/Workloads/,/SRDs/' | \ 
awk -v RS="" '{printf "%-20s %-10s %-20s
",$3,$9,$6}' | \ 
grep -v -E "###|SRDs" | grep -v -E ".OTHER"
```

This filter produces output similar to the following:

```bash
CMS # ./listwklds1.sh
Generating historical config report.
Please be patient. This may take several minutes.

<table>
<thead>
<tr>
<th>Workload</th>
<th>Type</th>
<th>Host</th>
</tr>
</thead>
<tbody>
<tr>
<td>va01sv.zko.hp.com</td>
<td>Whole</td>
<td>va01sv.zko.hp.com</td>
</tr>
<tr>
<td>vmhpux1</td>
<td>VM</td>
<td>va01sv.zko.hp.com</td>
</tr>
<tr>
<td>vmlin1</td>
<td>VM</td>
<td>va01sv.zko.hp.com</td>
</tr>
<tr>
<td>sales-va02cm</td>
<td>gWLM-created</td>
<td>va02cm.zko.hp.com</td>
</tr>
<tr>
<td>finance-va02cm</td>
<td>gWLM-created</td>
<td>va02cm.zko.hp.com</td>
</tr>
<tr>
<td>va00np00.zko.hp.com</td>
<td>Whole</td>
<td>va00np00.zko.hp.com</td>
</tr>
<tr>
<td>vmhpux2</td>
<td>VM</td>
<td>va01sv.zko.hp.com</td>
</tr>
<tr>
<td>legacy01</td>
<td>Whole</td>
<td>legacy01</td>
</tr>
<tr>
<td>va00np10.zko.hp.com</td>
<td>Whole</td>
<td>va00np10.zko.hp.com</td>
</tr>
<tr>
<td>legacy02</td>
<td>Whole</td>
<td>legacy02</td>
</tr>
<tr>
<td>vmlin1</td>
<td>VM</td>
<td>va01sv.zko.hp.com</td>
</tr>
<tr>
<td>va02cm.zko.hp.com</td>
<td>Whole</td>
<td>va02cm.zko.hp.com</td>
</tr>
<tr>
<td>legacy03</td>
<td>Whole</td>
<td>legacy03</td>
</tr>
</tbody>
</table>

CMS #
The remaining task is to remove whole OS workloads that are subdivided by Integrity VM, pset, or fss workloads. This can be accomplished with a filter such as the following:

```bash
CMS # cat ./listwklds2.sh
WKLDFILE=/tmp/gwlmreport.config.out
gwlmreport config | awk -v RS=" " '{printf "%s %s %s
%s",$1,$3,$6,$9}' | grep ^Workload " | awk '{printf "%s %s %s
", $2, $3, $4}' | grep -v "Other" > $WKLDFILE
for w in `cat $WKLDFILE | grep Whole | awk '{print $1}'`; do
cat $WKLDFILE | grep -v Whole | 
 awk ' BEGIN {WM=0} 
{if (match($2,"'$w'")) {WM=1; printf "%20s %10s
$20s
",$1,$3,"'$w'"})
END {if (WM==0) {printf "%20s %10s
"'$w'","Whole","'$w'"'))
}
done
rm $WKLDFILE:w
CMS #
```

This filter produces output similar to the following:

```bash
CMS # ./listwklds2.sh
Generating historical config report.
Please be patient. This may take several minutes.

  vmhpux1  VM  va01sv.zko.hp.com
  vmlin1   VM  va01sv.zko.hp.com
  vmhpux2  VM  va01sv.zko.hp.com
  vmwin1   VM  va01sv.zko.hp.com
  va00np00.zko.hp.com  Whole  va00np00.zko.hp.com
  legacy01 Whole legacy01
  va00np10.zko.hp.com  Whole  va00np10.zko.hp.com
  legacy02 Whole legacy02
  sales-va02cm gWLM-created va02cm.zko.hp.com
  finance-va02cm gWLM-created va02cm.zko.hp.com
  legacy03 Whole legacy03
CMS #
```

**Workload Utilization Information**

Capacity Advisor stores information about the utilization of workloads for planning and optimization. This data can also be used for chargeback purposes. The desired metric is typically CPU core usage, but memory can also be factored in.

This section concentrates on tools provided by Capacity Advisor because this component can be configured to collect utilization information about all workloads. The reporting feature of gWLM can also provide useful utilization information. However, gWLM reports only on workloads that it is actively managing. The set of gWLM managed workloads can be a subset of all workloads in the environment. For more information about gWLM reporting tools, see `resourceaudit` and `topborrowers` on `gwlmreport(1)`.

**GUI: Capacity Advisor Profile Viewer**
The Capacity Advisor component of VSE stores utilization data about workloads VSE for reporting and planning purposes. Sets of historical data are called Profiles. You can start the Profile Viewer GUI tool from HP SIM by clicking the Optimize tab and then selecting **Capacity Advisor**⇒**View Profile...**, or from the Virtualization Manager screen by clicking any of the utilization bar graphs.

The Profile Viewer displays the historical data for a workload, partition, or server. A useful feature for relating workloads to specific physical servers is the hierarchy selection box near the top of the viewer page, as shown in the following figure. Scroll down the hierarchy to specify a workload. This displays the utilization data for that workload, as shown in the following figure. The relevant data (Average) appears below the graph, under Interval Metric Summary, in the Absolute Utilization (CPUs) column.

You can select either CPU or memory utilization; in the following figure, CPU is selected. Along with the graph of utilization is a table of statistics that includes average utilization over the period of interest.
CLI: capreport tool

Capacity Advisor contains a command-line tool called capreport that allows historical utilization reports to be generated for workloads in the VSE environment. The capreport tool generates the report as a series of HTML files and subdirectories. By default, this output is in the current working directory from where the tool is run. In the following example, the tool is run in /tmp:

```
CMS # cd /tmp
CMS # capreport -t util -b 200709170000 -e 200709240000 -n va00np00
CMS # ls 2007-10-3-105329-612/CapAdReport
-C             en             images         js
va00np00.html
css            graphImages    index.html     report.tar
CMS #
```

The main output files are the index.html file and a file for each workload or partition covered in the report. The HTML file for the workload contains the average utilization value for that workload. You can view the report using a browser on the CMS, as shown in the following example:
The average utilization value in this HTML file can be extracted for use with the command-line script. The following is the source text for the preceding page:

CMS # more va00np00.html
<!-- saved from url=(0014)about:internet -->

<html>
<head><META http-equiv="Content-Type" content="text/html; charset=utf-8">
<meta content="text/html; charset=iso-8859-1" http-equiv="Content-Type">
</head>

<table>
<thead>
<tr>
<th>CPU</th>
<th>Absolute Utilization</th>
<th>Percent of Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>2.48CPUs</td>
<td>69.1%</td>
</tr>
<tr>
<td>Peak</td>
<td>3.56CPUs</td>
<td>89.0%</td>
</tr>
<tr>
<td>90th Percentile</td>
<td>3.51CPUs</td>
<td>87.7%</td>
</tr>
<tr>
<td>Max 15 Minute Sustained</td>
<td>3.52CPUs</td>
<td>87.9%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Memory</th>
<th>Absolute Utilization</th>
<th>Percent of Allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.87GB</td>
<td>5.5%</td>
</tr>
<tr>
<td>Peak</td>
<td>0.92GB</td>
<td>5.8%</td>
</tr>
<tr>
<td>90th Percentile</td>
<td>0.89GB</td>
<td>5.6%</td>
</tr>
<tr>
<td>Max 15 Minute Sustained</td>
<td>0.92GB</td>
<td>5.8%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Network I/O</th>
<th>Absolute Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.04Mb/s</td>
</tr>
<tr>
<td>Peak</td>
<td>0.35Mb/s</td>
</tr>
<tr>
<td>90th Percentile</td>
<td>0.04Mb/s</td>
</tr>
<tr>
<td>Max 15 Minute Sustained</td>
<td>0.29Mb/s</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Disk I/O</th>
<th>Absolute Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>0.03MB/s</td>
</tr>
<tr>
<td>Peak</td>
<td>0.12MB/s</td>
</tr>
<tr>
<td>90th Percentile</td>
<td>0.03MB/s</td>
</tr>
<tr>
<td>Max 15 Minute Sustained</td>
<td>0.05MB/s</td>
</tr>
</tbody>
</table>
The average Utilization parameter is contained in a table called `SummaryTable_cpuTable` in row0. The following is an example filter for extracting this value:

```bash
CMS # cat ./getutil.sh
cat $1 | awk
   '/SummaryTable_cpuTable.row0/,/SummaryTable_cpuTable.row1/' \
| grep "CPUs"
CMS #
```

This filter produces the following output:

```bash
CMS # ./getutil.sh va00np00.html
<td>2.48CPUs</td>
CMS #
```

Memory utilization can be extracted in a similar manner.
Workload Allocation Information

For nPar and vPar workloads, the CPU core utilization of the workload reported by Capacity Advisor is the combined user and system time as seen by the OS running in the partition. This utilization typically is less than 100% of the cores that VSE allocates to the partition.

GUI: Capacity Advisor Utilization Report

The Capacity Advisory Utilization Report includes a summary table for partitions of a complex physical server. The CPU Count column is actually the average number of physical CPU cores allocated to the partition over the reporting period, as shown in the following example:

In this example, the CPU count is not a whole number because the nPars are under the control of gWLM, which is dynamically activating and deallocating TiCAP processing.

CLI: capreport tool

As described earlier, the capreport tool creates a subdirectory and files of an XML report for the utilization of a collection of workloads. The index.html file for the report includes the average physical CPU core allocation for the workload in the CPU Count column. For example:

```plaintext
CMS  # pwd
    /tmp/2007-10-3-105329-612/CapAdReport
CMS  # more index.html
...

<caption class="checkboxCell">Historical utilization metrics shown as 15 minute sustained peaks over the reporting period.
</caption>

<tr class="captionRow">
<th colIndex="0">System Name</th>
<th colIndex="1">CPU</th>
<th colIndex="2">Memory</th>
<th colIndex="3">Available Period</th>
</tr>
```
<table>
<thead>
<tr>
<th>CPU Count</th>
<th>Memory Size</th>
<th>Type</th>
<th>Available Period</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.30 @ 1.50 GHz</td>
<td>15.93GB</td>
<td>nPar, SRD Member, Server, Application Discovery</td>
<td>9/17/07 12:00:00 AM EDT - 9/24/07 12:00:00 AM EDT</td>
</tr>
</tbody>
</table>

The following is a simple filter to extract this allocation information:

```bash
cat $1 | awk '/ReportIndexPage_systemTable.row0/,/End: ReportIndexPage_systemTable/' | 
grep GHz
```

This filter produces the following output:

```bash
cms # ./getalloc.sh index.html
td>3.30 @ 1.50 GHz<td>
```
Workload Resource Guarantees with gWLM

VSE allows IT to control the level of resources given to workloads using gWLM and Integrity VM. One aspect of this control is the ability to set guarantees. In many cases, servers are sized – and therefore the cost to IT is based – on these guaranteed levels, making this an attractive parameter to use in chargeback.

GUI: VSE Manager Workload page

Looking again at the Virtualization Manager **Workload** page, workloads managed by gWLM have a non-null Policy Name field, as shown in the following figure:

To view a page that shows the properties of the policy, click the Policy Name in the table entry for the workload. The following figure is an example of this page. This example shows a policy with Min Size and Max Size settings. In this case, the Min Size is a guaranteed level of resources provided to this workload.
In the next example, the policy has an Owned Size as well as a Min Size and Max Size:

![Policy Configuration Screen](image)

In such cases, the Owned Size value is the guaranteed amount of resources for workloads with this policy.

**CLI: gwlmreport config command**

Workloads that are being actively managed by gWLM are part of gWLM Shared Resource Domains (SRDs). The following example is a section of the `gwlmreport config` command output. The example includes a list of managed SRDs and the workloads that are in them.

```
CMS # gwlmreport config
...
SRD - va01sv.srd
Mode    : Managed
Status  : Deployed
Interval: 15 seconds
Workload - va01sv.zko.hp.com.OTHER
    Policy          : Owns_2-Max_4
    Compartment Type: hpvm
Workload - vmwin1
    Policy          : owns.5-max-remaining
    Compartment Type: hpvm
Workload - vmhpu2
    Policy          : Owns_1-Max_Remaining
    Compartment Type: hpvm
Workload - vmhpu1
    Policy          : Owns_1-Max_Remaining
    Compartment Type: hpvm
Workload - va00vm00.zko.hp.com
    Policy          : owns.5-max-remaining
    Compartment Type: hpvm

Listed with each workload is its policy. The last step is to determine the specific policy settings of workload’s policy. The following section of the `gwlmreport config` command output contains policy definition information:

```
CMS # gwlmreport config
...
#############
#### Policies
#############

Policy - Fixed_8
    Type: Fixed, Priority: 0, Ticap Enabled: No
    CPU [ Min: 8.0, Own: 8.0, Max: 8.0,
           Weight: 0.0, Target: 8.0, Convergence Rate: 1.0 ]

Policy - Owns_0.5-Max_1
    Type: OwnBorrow, Priority: 100, Ticap Enabled: No
    CPU [ Min: 0.1, Own: 0.5, Max: 1.0,
           Weight: 0.5, Target: 75.0, Convergence Rate: 0.0 ]

Policy - Owns_3-Max_6
    Type: OwnBorrow, Priority: 100, Ticap Enabled: No
    CPU [ Min: 1.0, Own: 3.0, Max: 6.0,
           Weight: 3.0, Target: 75.0, Convergence Rate: 0.0 ]

Policy - Fixed_4
    Type: Fixed, Priority: 0, Ticap Enabled: No
    CPU [ Min: 4.0, Own: 4.0, Max: 4.0,
           Weight: 0.0, Target: 4.0, Convergence Rate: 1.0 ]
...
```
In this output, note that the Owned value for the policy can be determined. In cases where the GUI reports only a minimum and maximum value, the `gwlmreport` output includes an Owned value that is equal to the minimum value. Note also that many more policies might be defined and listed than those shown to be actually in use.

The following simple script uses the output of `gwlmreport config` to first identify workloads that are part of SRDs, then to identify the matching policy and print its owned value:

```bash
CMS # cat ./wkldguargwlm.sh
GWLМCONFIGFILE=/tmp/gwlmreport.config.out
SRDWKLDSPFILE=/tmp/gwlmreport.config.srdwklds.out
WRKLDOWNEDFILE=/tmp/gwlmreport.config.wkldowned.out

gwlmreport config > $GWLМCONFIGFILE

cat $GWLМCONFIGFILE | awk '/SRDs/,/EOF/' | grep -v "####" | \ 
  grep -E "^ Workload|^ Policy" | sed 's/ - / /g' | sed 's/:/ /g'| \ 
  awk '{ if (match($1,"Workload")) {printf "%s ",$2} if \ 
    (match($1,"Policy")) {print $2}}' \ 
  > $SRDWKLDSPFILE

cat $GWLМCONFIGFILE | awk '/Policies/,/Workloads/' | grep -v "####" | \ 
  awk -v RS=" " '{printf "%s %s
", $3,$16}' | sed 's/,// ' > $WRKLDOWNEDFILE

for w in `cat $SRDWKLDSPFILE | awk '{print $1}'`; do
  printf "%s 
  WKLDPLCY=`cat $SRDWKLDSPFILE | grep $w | awk '{print $2}'`
  grep $WKLDPLCY $WRKLDOWNEDFILE | awk '{print $2}'
done

rm $GWLМCONFIGFILE $SRDWKLDSPFILE $WRKLDOWNEDFILE
CMS #
```

This script produces the following output:

CMS # ./wkldguargwlm.sh
Generating historical config report.
Please be patient. This may take several minutes.
va00np10.zko.hp.com 1.0
va00np00.zko.hp.com 1.0
sales-va02cm 0.5
va02cm.zko.hp.com.OTHER 1.0
finance-va02cm 0.5
va01sv.zko.hp.com.OTHER 2.0
vmwin1 0.5
vmhpux2 1.0
vmhpux1 1.0
va00vm00.zko.hp.com 0.5
CMS #
Workload Resource Guarantees with Integrity VM

For Integrity VM guests that are not being managed by gWLM, the amount of guaranteed CPU is known as an entitlement. The net amount of guaranteed physical core processing is the entitlement percentage multiplied by the number of virtual CPUs (vCPUs) of the virtual machine.

GUI: Integrity VM Manager

You can use the Integrity VM Manager tool to obtain information about virtual machines. In the following view, the summary properties of a virtual machine are displayed, including entitlement and vCPUs. In this example, the entitlement for the virtual machine is 40% of 4 CPU cores.

CLI: `hpvmstatus` run on Integrity VM Host CMS using HP SIM mxexec
The `hpvmstatus` command runs on a specific Integrity VM Host server and shows the summary properties of the virtual machines on that server. In the following example, the `-r` flag specifically shows the vCPUs and entitlement values:

```
HPVM HOST # hpvmstatus -r
[Virtual Machine Resource Entitlement]
[Virtual CPU entitlement]

<table>
<thead>
<tr>
<th>Virtual Machine Name</th>
<th>VM #</th>
<th>Entitlement</th>
<th>Maximum</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>vmlin1</td>
<td>1</td>
<td>80.0%</td>
<td>100.0%</td>
<td>0.3%</td>
</tr>
<tr>
<td>1743854</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmhpux1</td>
<td>2</td>
<td>40.0%</td>
<td>100.0%</td>
<td>1.3%</td>
</tr>
<tr>
<td>127576469</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmhpux2</td>
<td>3</td>
<td>40.0%</td>
<td>100.0%</td>
<td>3.0%</td>
</tr>
<tr>
<td>45138177</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>vmwin1</td>
<td>4</td>
<td>40.0%</td>
<td>100.0%</td>
<td>0.6%</td>
</tr>
<tr>
<td>3052306</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

The following script converts the virtual machine entitlement and vCPU values to an aggregate amount of guaranteed physical core processing:

```
HPVM HOST # cat wkldguarhpvm.sh
hpvmstatus -r | \
awk '{ if (NR >5) printf "%s %2.2f\n",$1,$3*($4/100)}'
```

This script produces the following output:

```
HPVM HOST # ./wkldguarhpvm.sh
vmlin1 0.80
vmhpux1 1.60
vmhpux2 1.60
vmwin1 0.80
```

To run this command from the CMS command line on remote Integrity VM systems that are not managed by gWLM, you can use the HP SIM custom tool and `mxexec` commands.
Listing Physical Servers

This section describes how to identify the physical servers in the VSE environment. A feature of HP SIM is its ability to organize and present a vast array of different systems for management and monitoring. It also includes optional features to allow the user to add information for asset tracking. It provides both GUI and CLI interfaces. Virtualization Manager integrates into HP SIM and adds the ability to graphically depict the relationships of servers and workloads as well as access to other VSE GUI tool interfaces.

GUI: HP SIM VSE Manager All VSE Resources page

As shown in the following figure, the VSE Manager All VSE Resources page displays a complete hierarchical view of the system, from workloads to physical servers.

All of the physical servers are indicated as the highest level of nesting. In this example, the managed VSE nodes consist of six physical systems that are grouped as follows:
- va00-cplx_USE4452HL6 is a complex server divided into two nPar workloads.
- va01sv is a noncomplex server with 4 VM guest workloads running on it. It also includes the VSE-created Other workload.
- va02cm is a noncomplex server with two user-defined fss workloads and the additional VSE-created Other fss workload.
- legacy01, legacy02, and legacy03 are noncomplex servers with whole OS workloads.

**CLI: mxnode and mxquery commands**

HP SIM keeps many parameters about systems it has detected in its database. The `mxnode` command provides a simple tool to list and obtain basic data about servers. The following example command, which includes the `-ln` flag, displays a simple listing of the servers that are managed nodes from a HP SIM perspective:

```bash
CMS # mxnode -ln
va01sv
va03sa
legacy01
va00mp
va00vm00
va00vm03
va00-cplx_USE4452HL6
va00vm01
va02cm
va00vm02
legacy03
legacy02
va00np00
va00np10
CMS #
```

The next step is to determine which servers on this list are physical HP 9000 and Integrity servers. The output from the `mxnode` command output for an individual server includes parameters that can help with this identification. The `mxnode -ld` command displays the following detailed data for a specific server:

```bash
CMS # mxnode -ld va01sv
System name: va01sv
Host name: va01sv.zko.hp.com
IP addresses: 16.116.90.85
OS name: HPUX
OS revision: 11.23
OS revision text: HP-UX B.11.23 U
OS vendor: HP
...
Device type: Server
Device subtype: HPVMHost
Device subtype: HPIntegrity
Device subtype: SMI
Device subtype: Storage
Model: ia64 hp server rx4640
...
CMS #
```

For a physical HP 9000 or Integrity server, the Device type and Device subtype fields have the following values:
- For a complex (cell-based) physical server
  - Device type: complex
- For all other (noncomplex/non-cell-based) physical servers
  - Device type: Server
  - Device subtype: HP Integrity or Device subtype: HP9000
  - None of the following entries
    - Device subtype: nPar
    - Device subtype: vPar
    - Device subtype: HPVMGuest

The following is a simple shell loop that extracts all node records from the `mxnode` output and looks for physical servers:

```bash
CMS # cat ./listphys.sh
for i in `mxnode -ln`; do
  printf "\n%s\n" $i
  mxnode -ld $i | grep "^Device" | grep "type"
done | awk -v RS=" " '{
    if (match($0,"Complex")) (printf "%s Complex\n", $1) 
    if (match($0,"Integrity|HP9000"))
    { if (!match($0,"vPar|nPar|HPVMGuest")){printf "%s Non-Complex\n", $1})
  }'
CMS #
```

This script produces the following output:

```
CMS # ./listphys.sh
va01sv Non-Complex
legacy01 Non-Complex
va00-cplx_USE4452HL6 Complex
va02cm Non-Complex
legacy03 Non-Complex
legacy02 Non-Complex
CMS #
```

As an alternative to `mxnode` command, you can use the HP SIM `mxquery` command. The `mxquery` command presents a number of more sophisticated queries to HP SIM that include access to the same collections that are presented in the graphical tool. This command also allows the user to create new queries. Running `mxquery` with no arguments gives a list of built-in queries that are available, as shown in the following example:

```
CMS # mxquery
All Clusters
MSCS Clusters
OpenVMS Clusters
...  
All Systems
All Servers
...  
All HP Service Events
All nPartition Servers
All HP Integrity Virtual Machines
All Virtual Partition Servers
All Resource PartitionsAll Shared Resource Domains
All HP Serviceguard Clusters
All Standalone Servers
All p-Class Racks
All e-Class Enclosures
```
All c-Class Racks
All HP Integrity VM Guests
All nPartitions
All Virtual Partitions
Events older than 90 days

From a HP SIM perspective, the All Systems output can encompass many different types of devices that HP SIM can reach on the network. Therefore, HP SIM might return many entries that do not relate to VSE servers. Similarly, HP SIM’s multiplatform support means that All Servers might return more than just the VSE managed physical servers that are of interest.

To find physical servers that are complexes, add the “All nPartition Servers” query and the –e flag to the mxquery command. For example:

```
CMS # mxquery -e "All nPartition Servers"
```

This filter produces the following output:

```
DeviceName: va00-cplx_USE4452HL6
```

To display just the complex names, use the following simple filter:

```
CMS # cat listcmplx.sh
mxquery -e "All nPartition Servers" | grep "^DeviceName" | awk '{printf "%s Complex\n",$2}'
```

To display noncomplex HP 9000 and HP Integrity servers, you must create a new query using “All Standalone Servers” as a template. The mxquery –lf command lists the definition of the query, as shown in the following example:

```
CMS # mxquery -lf "All Standalone Servers"
```

```xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<query-list>
  <query id="61" name="All Standalone Servers" type="DeviceViews" category-name="HPVSE" owner="mxadmin">
    <criteria name="CriteriaByProductType" sense="INCLUDE">
      <value>Server</value>
    </criteria>
    <criteria name="SubType" sense="INCLUDE">
      <value>HP9000</value>
      <value>HPIntegrity</value>
    </criteria>
  </query>
</query-list>
```
The highlighted "INCLUDE" and "EXCLUDE" flags indicate which kinds of servers match this query. Notice that this query lists all noncomplex physical servers except those that are also VM Host servers. A custom query that does not exclude these servers is easy to create. First copy the existing definition of the "All Standalone Servers" query into a file with a new name. For example:

```bash
CMS # mxquery -lf "All Standalone Servers" > /tmp/physicalservers.mxquery
CMS #
```

Then edit this definition so that VM Hosts are not excluded:

```bash
CMS # vi /tmp/physicalservers.mxquery
<?xml version="1.0" encoding="ISO-8859-1"?>
<query-list>
  # (changed the query name from "All Standalone Servers"
  # to "All Physical Servers")
  <query id="61" name="All Physical Servers" type="DeviceViews" category-name="HPVSE" owner="mxadmin">
    <criteria name="CriteriaByProductType" sense="INCLUDE">
      <value>Server</value>
    </criteria>
    <criteria name="SubType" sense="INCLUDE">
      <value>HP9000</value>
      <value>HPIntegrity</value>
    </criteria>
    <criteria name="SubType" sense="EXCLUDE">
      # (deleted "<value>HPVMHOST</value> line)
      <value>vParMonitor</value>
      <value>nPar</value>
      <value>vPar</value>
      <value>HPVMGuest</value>
    </criteria>
  </query>
</query-list>
~
:wq
CMS #
```

The new query can then be registered using the -a -f flags to the mxquery command:

```bash
CMS # mxquery -a -f /tmp/physicalservers.mxquery
CMS #
```

Finally, the new query can be run and filtered to provide a list of noncomplex physical servers:

```bash
CMS # cat listnoncmplx.sh
mxquery -e "All Physical Servers" | \
```
This procedure produces the following output:

```
CMS # ./listnoncmplx.sh
va02cm Non-Complex
va01sv Non-Complex
...
CMS #
```
Workload and Physical Server Associations

Earlier sections of this paper demonstrate how to list workloads and physical servers in the environment. This section describes how to combine the information to determine the physical server of a workload. This is straightforward with the GUI tools but requires some filters to put into a form that is suitable in a larger script.

GUI: VSE Manager All VSE Resources page

As described earlier, the VSE Manager All VSE Resources page shows the complete hierarchy. For example:
CLI: gwlmreport config tool

As described earlier, the output of the gwlmreport config tool can be used to list all workloads. A filter can be used to remove all but the end “leaf” workloads of the hierarchy that are associated with end-user processing. This output contained a line for each workload with the workload’s name, type, and host:

```
CMS # ./listwklds2.sh
Generating historical config report.
Please be patient. This may take several minutes.
  vmhpux1   VM    va01sv.zko.hp.com
  vmlin1    VM    va01sv.zko.hp.com
  vmhpux2   VM    va01sv.zko.hp.com
  vmwin1    VM    va01sv.zko.hp.com
  va00vm03.zko.hp.com Whole va00vm03.zko.hp.com
  va00np00.zko.hp.com Whole va00np00.zko.hp.com
  legacy01 Whole legacy01
  va00np10.zko.hp.com Whole va00np10.zko.hp.com
  legacy02 Whole legacy02
  sales-va02cm gWLM-created va02cm.zko.hp.com
  finance-va02cm gWLM-created va02cm.zko.hp.com
  legacy03 Whole legacy03
CMS #
```

The next task is to determine either the physical server or complex with which the host of the workload is associated. The previous example used the HP SIM mxnode command to display the properties of HP SIM managed nodes and determine their type. For example:

```
CMS # mxnode -1d va01sv
System name: va01sv
Host name: va01sv.zko.hp.com
IP addresses: 16.116.90.85
OS name: HPUX
OS revision: 11.23
OS revision text: HP-UX B.11.23 U
OS vendor: HP
...
Device type: Server
Device subtype: HPVMHost
Device subtype: HPIntegrity
Device subtype: SMI
Device subtype: Storage
Model: ia64 hp server rx4640
...
CMS #
```

You can use the gwlmreport config and mxnode commands together to identify the workload hosts, as shown in the following script:

```
CMS # cat wkldphys.sh
WKLDSFILE=/tmp/gwlmreport.config.out
WKLDSFLTRD=/tmp/wkldsfltrd

# extract a list of all workloads
```
This script produces the following output:

In this example, all but two of the workloads are running on noncomplex physical servers. The remaining two workloads are nPars that share the same complex.
Physical Server Serial Numbers and Asset Tags

If an organization is interested in the relationship of physical servers to cost, it can be useful to provide more than just a server name in resource allocation data. HP SIM discovers and maintains server serial numbers and provides a field for customers to use for an optional asset tag.

**GUI: HP SIM System page**

The HP SIM **System** page for a managed node displays the serial number for a physical server, as shown in the following example for a complex:

```
va00-cplx_US3E4452HL6 (Complex)
```

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complex Name</td>
<td>va00-cplx_US3E4452HL6</td>
</tr>
<tr>
<td>Serial Number</td>
<td>USE3E4452HL6</td>
</tr>
<tr>
<td>UUID</td>
<td>223, 158, 114, 16, 111, 254, 17, 217, 168, 74, 113, 115, 215, 162, 117, 167</td>
</tr>
<tr>
<td>Product Number - Current</td>
<td>AB204A</td>
</tr>
<tr>
<td>Product Number - Original</td>
<td>AB204A</td>
</tr>
<tr>
<td>Complex Profile Revision</td>
<td>1.0</td>
</tr>
</tbody>
</table>
```

The Asset Tag field is available in the Data Collection Report, which is located on the **Tools & Links** tab of the **System** page. The following is an example of this page from a noncomplex physical server for which the asset tag has been set by the IT organization. By default, this field is typically empty.
CLI: mxnode and mxreport commands

For complex physical servers, the output from the mxnode command contains both the asset tag and serial number. In the following example, the -ld flag is used with the mxnode command to obtain this information for a node that is known to be a complex:

```
CMS # cat ./cmplxsn.sh
mxnode -ld $1 | \n  grep -E "CreatorSerialNumber|^Asset"
CMS #
```

This command produces the following output:

```
CMS # ./cmplxsn.sh va00-cplx_USE4452HL6
CreatorSerialNumber: USE4452HL6
Asset number: 0000
CMS #
```

For noncomplex physical servers, the mxnode command output does not contain serial number information, so another tool is required. Use the mxreport command to get the serial numbers for this type of system. The mxreport utility creates reports based on data in the HP SIM database. You can run any of several predefined reports, or you can use additional, user-defined reports. The following mxreport command uses the -l –x flags to list all available reports:

```
CMS # mxreport -l -x
Inventory - Servers
Inventory - Clients
Installed Controllers - Servers
Installed Controllers - Clients
Operating System Information - Servers
Operating System Information - Clients
CPU - Servers
CPU - Clients
Physical Disk Drives - Servers
Logical Disk Drives - Servers
...
CMS #
```

The “Inventory – Servers” report displays serial numbers and asset tags for noncomplex servers. The following example shows its use with two noncomplex servers:

```
CMS # mxreport -n -S va01sv,va02cm -e "Inventory - Servers" -x CSV

Inventory - Servers
Associated systems: va01sv,va02cm
Report date and time: Monday, October 1, 2007 4:05:10 PM EDT
Inventory
"System Name","Product Type","Product Model","Serial Number","Asset Tag","Memory Size (KB)","Operating System Name","Operating System Vendor","Location","System Owner"
```
Physical Server CPU Cores and Memory

As described earlier, IT might want to allocate the costs of physical servers across its workloads proportionally. If this is the case, then you must first determine the CPU core and memory configuration of the physical server.

GUI: HP SIM System page

The HP SIM System page for a physical server can be used to obtain component information for servers. For complex physical servers, to view the number of CPU cores and amount of memory, expand the Summary of Components heading on the main System page, as shown in the following example:

For noncomplex physical servers, you can view the information in the Data Collection Report, under the Tools & Links tab of the System page. (The asset tag is also displayed on this page.) The following example page shows component information for noncomplex physical servers. Under the CPU section, there is one line per physical CPU core in the server.
CLI: mxquery and mxreport commands

CPU Cores

For noncomplex physical servers, the `mxreport` command produces a report that shows the number of CPU cores. The “CPU – Servers” query lists all CPU cores for all noncomplex servers with one line of output per CPU core. For example:

```
CMS # mxreport -e "CPU - Servers" -x CSV | more
CPU - Servers
Associated systems: All Servers
Report date and time: Monday, October 1, 2007 5:25:56 PM EDT
CPU
"System Name" ,"CPU Type" ,"CPU Speed (MHz)" ,"Slot Number"
"va00np00" ,"Intel(R) Itanium(R) 2 Processor" ,"1500" ,"3"
"va00np00" ,"Intel(R) Itanium(R) 2 Processor" ,"1500" ,"1"
"va00np00" ,"Intel(R) Itanium(R) 2 Processor" ,"1500" ,"2"
"va00np00" ,"Intel(R) Itanium(R) 2 Processor" ,"1500" ,"0"
...
CMS #
```

A simple filter can show the exact number of cores for a specific noncomplex server. The following example obtains the number of cores for a server named va02cm:

```
CMS # mxreport -e "CPU - Servers" -x CSV | grep va02cm | wc -l
2
CMS #
```

Obtaining the number of CPU cores for a complex is more complex. The CPU core count is stored in HP SIM for each partition of the complex, not as a single value for the complex as a whole. The following three commands can be used:

```
- mxquery –e “All nPartition Servers”: Lists all complexes. DeviceName is the name of the complex. For example:

CMS # mxquery –e "All nPartition Servers" | more
...
NoticeID: 226
HWStatus: 0
MPStatus: 0
SWStatus: 0
```
mxquery –e “All nPartitions”: Lists all partitions of all complexes. DeviceName is the name of the partition, and AssociatedDeviceName is the name of the complex to which it belongs. For example:

```
CMS # mxquery -e "All nPartitions" | more
```

mxreport –e “CPU – Servers” –x CSV: Lists all CPU cores of all servers and partitions (but not complexes directly). It includes one line of output per CPU core, with the name of the partition included as a field. For example:

```
CMS # mxreport -e "CPU - Servers" -x CSV | more
```

### CPU - Servers

Associated systems: All Servers

Report date and time: Monday, October 1, 2007 5:25:56 PM EDT

CPU

"System Name","CPU Type","CPU Speed (MHz)","Slot Number"
"va00np00","Intel(R) Itanium(R) 2 Processor","1500","3"
The following is an example of a simple script that issues the three commands and processes the output to count CPU cores in a complex:

```bash
CMS # cat cmplxcores.sh
for i in `mxquery -e "All nPartition Servers" | grep "^DeviceName" | awk '{print $2}'`; do
    printf "%s " $i
    let CCPUS=0
    for n in `mxquery -e "All nPartitions" | awk -v RS=" " '{ printf "%s %s
",$34,$12}' | grep $i | awk '{print $2}'`; do
        let NCPUS=`mxreport -e "CPU - Servers" -x CSV | grep $n | wc -l`
        let CCPUS="$CCPUS + $NCPUS"
        done
    done
    print "$CCPUS"
done
CMS #
```

This script produces the following output:

```bash
CMS # ./cmplxcores.sh
va00-cplx_USE4452HL6 8
CMS #
```

Memory

The `mxreport "Inventory - Servers"` command includes a line for every server, with a field in the output for the amount of physical memory in that server. You can include the names of specific physical servers on the command line. For example:

```bash
CMS # mxreport -e -n -S va01sv(va02cm) "Inventory - Servers" -x CSV
```

Inventory - Servers

Associated systems: va01sv,(va02cm)

Report date and time: Tuesday, October 2, 2007 1:50:28 PM EDT

Inventory

"System Name","Product Type","Product Model","Serial Number","Asset Tag","Memory Size (KB)","Operating System Name","Operating System Vendor","Location","System Owner"
"va01sv","Server","ia64 hp server rx4640","USE4452N2H","0001","33541676","HP-UX","HP","",""
"va02cm","Server","ia64 hp server rx2600","US40576064","0002","8377064","HP-UX","HP","",""
CMS #
The `mxreport "Inventory - Servers"` command also provides the same information about partitions of a complex physical server. To find the total memory for a complex, the amount in each partition must be added together. The following example script uses the `mxquery "All nPartition Servers", mxquery "All nPartitions" and mxreport "Inventory - Servers" commands to report the amount of memory for complexes in the environment. For example:

```bash
CMS # cat ./cmplxmem.sh
for i in `mxquery -e "All nPartition Servers" | grep "^DeviceName" | awk '{print $2}'`; do
  printf "%s " $i
  let CMEM=0
  for n in `mxquery -e "All nPartitions" | awk -v RS="" '{printf "%s %s\n",$34,$12}' | grep $i | awk '{print $2}'`; do
    let NMEM=`mxreport -e -n -S $n "Inventory - Servers" -x CSV | sed 's/"/\"/g' | grep "^$n" | awk -v FS=""," '{print $6}'`
    let CMEM="$CMEM + $NMEM"
  done
  print $CMEM
done
CMS #
```

This procedure produces the following output:

```bash
CMS # ./cmplxmem.sh
va00-cplx_USE4452HL6 33413528
```

50
Physical Server iCAP

The iCAP and TiCAP of HP’s Utility Pricing Solutions for HP Integrity and HP 9000 servers allow customers flexibility in how they pay for and share computing resources. When iCAP and TiCAP are used, you must gather additional information from VSE for chargeback.

In iCAP configurations, the number of physical CPU cores and the amount of physical memory in a server might exceed the number and amount that the customer is entitled to use through purchased usage rights. Customers using iCAP can install either cores or cells containing both cores and memory in their servers, but they are not required to purchase usage rights for them at that time. In these cases, for chargeback purposes, IT might be more interested in a workload’s utilization relative to the pool of resources with usage rights rather than relative to what is physically installed.

A simplification in gathering data for chargeback is that currently iCAP is only supported on cell-based complex physical servers. This means that these parameters only need to be determined for that type of physical server.

The main reporting tool for iCAP and TiCAP is the CLI `icapstatus` command. This command runs on the VSE managed node where iCAP is being used. If the server is partitioned, the output when run on any of the partitions provides the count of components with usage rights and those without for the entire server.

GUI: CLI `icapstatus` command via HP SIM Custom Tools mechanism

HP SIM includes a mechanism for the GUI user to create new tool definitions that can be invoked from the GUI. The mechanism allows the tool to run commands on remote systems that are managed by VSE. Using this feature, a tool that runs the `icapstatus` command on complex physical servers can display the results within the HP SIM GUI.

To create the new tool from within the HP SIM GUI, select the Tools tab, then select Custom ToolsÆNew Custom Tools. Now enter the following input to these screens:

- Step 1: Select the tool to create.
  - Click Remote tool - …
- Step 2: Describe how the new custom tool will work.
  - At a minimum, select or define:
    - “Name of the tool, …”: User defined, for example: “remote icapstatus”
    - “The user account on the target system..”: Choose “Special user (“root”)…”
    - “Command with parameters: …”
      - Enter `/usr/sbin/icapstatus`

By default, the tool appears as names under Custom Tools on the Tools tab. If changes to the tool are required, you can make edits on the Tools tab by selecting Custom ToolsÆManage Custom Tools.

Once the tool is created, you can start it from the Tools tab by selecting Custom ToolsÆ“remote icapstatus” (or the specific name you gave it in the preceding steps). The tool takes you through the following screens:

- Step 1: Select Target Systems
  - The HP SIM collection “All nPartitions” provides a list of all the partitions in all partitionable complex physical servers. The `icapstatus` command needs to run in only one partition in each complex. Select at least one partition from each of the complexes in the environment.
- Step 1: Verify Target Systems
  o Select the systems you selected in the previous screen.
- Step 2: Task Confirmation
  o Click Run Now at the lower right.
- Task Results
  o If the tool definition and system selection are correct, the tool runs properly and the “Status:” field indicates “Complete”.
  o The Stdout tab under Target Details displays the output of the remotely run icapstatus command, as shown in the following example screen:

```
COMPLEX PARTITION # icapstatus
Software version: B.08.01
System ID: va00np00
Serial number: USE4452HL6
Product number: AB204A
Unique ID: e99e7210-6ffe-11d9-a64a-7173d7a275a7
System contact e-mail: Not set
...
Instant Capacity Resource Summary
```
Number of cells without usage rights: 0
Number of inactive cells: 0

**Amount of memory without usage rights:**

<table>
<thead>
<tr>
<th>GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
</tr>
</tbody>
</table>

Amount of inactive memory: 0.0 GB

**Number of cores without usage rights:**

<p>| |</p>
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Number of inactive cores: 4
Number of cores using temporary capacity: 0
Temporary capacity available: 101 days, 17 hours, 0 minutes
Projected temporary capacity expiration: N/A

### Allocation of Instant Capacity Resources among the nPartitions

<table>
<thead>
<tr>
<th>Intended Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>nPar</td>
</tr>
<tr>
<td>ID</td>
</tr>
<tr>
<td>====</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>N/A</td>
</tr>
</tbody>
</table>

The output fields include the amount of memory and CPU cores that do not have usage rights. In the example, no memory is without usage rights, but 4 CPU cores do not have usage rights.

The HP SIM `mxexec` tool can be used to run tools defined through the Custom Tools interface on remotely managed systems from the CMS command line. In the previous section, a Custom Tool named “remote icapstatus” was created using the HP SIM GUI. To confirm that this Custom Tool exists, enter the `mxtool` command with the `-ln` flag. For example:

```
CMS # mxtool -ln | grep "remote icapstatus"
remote icapstatus
CMS #
```

To run the Custom Tool through the `mxexec` tool on partition va00np00 of a remote complex physical server, enter the following command:

```
CMS # mxexec -t "remote icapstatus" -n va00np00
Running tool remote icapstatus with job id 46597.
Task Name :defRunNowTaskId_1191333230487_39
Job ID :46597
Tool Name :remote icapstatus
Job State :Complete
User Name :root
Execute As User :<root-admin/>
Start Time :Tuesday, October 2, 2007 9:53:50 AM EDT
End Time :Tuesday, October 2, 2007 9:53:51 AM EDT
```
An earlier section of this paper provided an example showing how to determine the CPU cores in a server. The following example extends that script by using `mxexec`, a Custom Tool, and `icapstatus` to also display the number of cores with usage rights. For example:

```bash
CMS # cat ./cmplxcores_icap.sh
for i in 'mxquery -e "All nPartition Servers" | grep "^DeviceName"'
    awk '{print $2}';
for n in 'mxquery -e "All nPartitions" | awk -v RS="" '{
    printf "%s %s
",$34,$12}';
let CCPUS=0
for n in 'mxexec -t "remote icapstatus" -n $n | grep "Number of cores without usage rights"'
    awk '{print $7}';
let LCPUS="$CCPUS - $ULCPUS"
printf " Physical Cores %d Cores with usage rights %d\n"
$CCPUS $LCPUS
done
ULCPUS="mxexec -t "remote icapstatus" -n $n | grep "Number of cores without usage rights" | awk '{print $7}';
let LCPUS="$CCPUS - $ULCPUS"
printf " Physical Cores %d Cores with usage rights %d\n"
$CCPUS $LCPUS
done
CMS #
```

This command produces the following output:

```
CMS # ./cmplxcores_icap.sh
va00-cplx_USE4452HL6  Physical Cores 8 Cores with usage rights 4
CMS #
```

Similarly, the earlier example script that determined the physical memory of a complex can also be modified to report the amount of memory in the complex that has usage rights:

```
CMS # cat cmplxmem_icap.sh
```
for i in `mxquery -e "All nPartition Servers" | grep "^DeviceName`;
    do
        printf "%s " $i
    let CMEM=0
    for n in `mxquery -e "All nPartitions" | awk -v RS=" " '{
            printf "%s %s\n",$34,$12}' | grep $i | awk '{print $2}'`; do
        let NMEM=`mxreport -e -n -S $n "Inventory - Servers" -x CSV | sed 's/"//g' | grep "^$n" | awk -v FS="," '{print $6}'`
            let CMEM="$CMEM + $NMEM"
        done
        let ULMEM=`mxexec -t "remote icapstatus" -n $n | grep "Amount of memory without usage rights" | awk '{print $7}'`
        let CMEM="$CMEM/(1024*1024)"
        let LMEM="$CMEM - $ULMEM"
        printf " Physical Mem %d GB Mem with usage rights %d GB\n" $CMEM $LMEM
    done
CMS #

This command produces the following output:

CMS # ./cmplxmem_icap.sh
va00-cplx_USE4452HL6  Physical Mem 31 GB Mem with usage rights 31 GB
CMS #
Physical Server TiCAP

If a complex server configuration uses TiCAP as well as iCAP, additional cores without usage rights in a server can be activated and deactivated using separately purchased temporary capacity. TiCAP activations of cores represent a change in the pool of resources a workload is drawing on and, therefore, might be important to computing the relative utilization of a workload. In addition, because temporary capacity is a purchased separately from the server, IT organizations might want to specifically track its use in a chargeback implementation.

Using TiCAP and iCAP can simplify data gathering for chargeback purposes because both tools are supported only on cell-based, complex physical servers. Therefore, these parameters need to be determined only for that type of physical server.

The output of the `icapstatus` command has many parameters that govern the use of TiCAP in a complex. Unfortunately, even if the tool reports that no cores are currently using TiCAP and that the balance of TiCAP processing pool is zero, TiCAP processing might still have been used in the previous period of interest for reporting resource allocation.

**CLI: contents of /var/adm/icap.log on nPars and vPars**

One way to determine whether TiCAP was used over a historical period is to examine the TiCAP software logs on the partitions of the complex. Note that you must examine each log on all nPars or vPars. The log file resides in the following location:

```
COMPLEX PARTITION # 11 /var/adm/icap.log
-rw-r--r-- 1 root sys 444385 Oct  2 14:00 /var/adm/icap.log
COMPLEX PARTITION #
```

This file includes entries of the following format for every 30 minutes of TiCAP processing that occurs. For example:

```
Date:                                           10/02/07 02:30:04
Log Type:                                       Temporary capacity
debit
Temporary Capacity minutes debited:             30
Number of cores with usage rights:              4
Number of active cores:                         5
nPar    Active
ID      cores
  0      4
  1      1
Temporary capacity available:                   101 days, 18 hours, 30 minutes
```

The existence of at least one of these records in the reporting period indicates that TiCAP processing was consumed. The following is a simple filter to obtain the date of every debit of capacity:

```
COMPLEX PARTITION # cat icapdebits.sh
cat /var/adm/icap.log | \
awk -v RS="" '{printf "%s %s %s %s
",$2,$3,$6,$7,$8}' | \
grep "Temporary capacity debit"
COMPLEX PARTITION #
```
This script produces the following output:

```
COMPLEX PARTITION # ./icapdebits.sh
09/10/07 19:30:04 Temporary capacity debit
09/10/07 20:00:04 Temporary capacity debit
09/10/07 20:30:04 Temporary capacity debit
09/10/07 21:00:04 Temporary capacity debit
09/10/07 21:30:05 Temporary capacity debit
09/10/07 22:00:04 Temporary capacity debit
09/10/07 22:30:04 Temporary capacity debit
09/10/07 23:00:04 Temporary capacity debit
09/10/07 23:30:04 Temporary capacity debit
09/11/07 00:00:04 Temporary capacity debit
...
10/02/07 00:00:04 Temporary capacity debit
10/02/07 00:30:05 Temporary capacity debit
10/02/07 01:00:04 Temporary capacity debit
10/02/07 01:30:04 Temporary capacity debit
10/02/07 02:00:05 Temporary capacity debit
10/02/07 02:30:04 Temporary capacity debit
COMPLEX PARTITION #
```

You can create a more complex filter to look for records within a particular date range. The final script can be configured as an HP SIM Custom Tool and can be run from the CMS, either graphically or using the `mxexec` tool on every nPar and vPar in the complex.
Summary

IT organizations that require resource allocation data for chargeback can extract that information from the HP VSE environment using the many GUI and CLI tools provided. This paper describes the parameters that might be of interest and provides examples of how to use specific tools to determine their usefulness for a chargeback reporting period. IT organizations that want to automate the collection of resource allocation data can use the example command sequences to build larger scripts that suit their specific needs.
For more information

For more information about HP VSE virtualization technologies, visit the VSE website at www.hp.com/go/vse.