Making your HP Glance Pak perform

by Doug Grumann
Hewlett-Packard

Automobiles can function without their speedometers and other instruments, but most drivers prefer to have them in working order. Likewise, computers can function without tools to monitor their performance, but many system administrators find them essential. If you have ample budget to continually purchase new servers and endlessly expand your data center, perhaps you really don't need to worry about system performance. Most of us aren't in that situation, however. The HP system performance tools Glance and Performance Agent are often considered "standard equipment" for servers, because of their proven track record as useful management software. For system admins and data center managers charged with continually doing more with less, these tools are essential.

So what are these tools all about? Simply put, Glance is a management software product that lets you know what's running on your unix servers right now, by generating metrics that measure system resources as well as process and application data. It is a "deep dive diagnostic" for system performance. The Performance Agent takes basically the same metric set and logs the data for later export or analysis. Performance Agent (PA) can also generate alarms on complex metric conditions into other tools such as Operations Manager. Both Glance and PA bundled together are sold as the Glance Pak. You may have these products on your systems but not know their full capabilities or how to customize them to suit your needs best. This paper is not meant as an introduction for someone who has not heard of these products. Rather, I'll assume that you have heard of Glance and PA, and would like to know more about their configuration, use, and best practices, with an emphasis on how you can customize them.

Topics covered below include Context, Product Structure, Glance overview, Performance Agent overview, CPU scenario, Symptoms/alarms, Metrics, Memory Scenario, Multiprocessor and Virtualized systems, Managing collection, Importing and Exporting data, and some miscellaneous Tips and Tricks. Enjoy!

Context – "how does this relate to other management products?"

These days, keeping up with all the names of management software can be somewhat of a challenge in itself. There are several different offerings from HP. Both Glance and the Performance Agent fit into the Operations Center suite of HP-Software products. Glance and PA complement and enhance the performance drilldown capabilities of Operations Manager (which is also known as OVO). The Performance Manager (PM) is a centralized analysis tool that connects to and graph the data stored by PA.

Other analysis products include HP Performance Insight and Reporter, which connect and pull the metrics logged by PA into centralized databases. These products provide web-based graphical reporting capabilities for the historical system performance data

generated by PA. The Glance Pak products collect some of the same metrics that one can obtain via SiteScope monitors, but unlike remote probes, Glance and PA run on the target system that is being monitored. This allows the Glance and PA to get very detailed performance metrics with high frequency at a minimal overhead.

Products like Network Node Manager concentrate on networking as opposed to server analysis. HP management software products related to the Business Availability Center (BAC) suite focus on specific application and service monitoring. There are other HP-Software products that focus on Provisioning, Change Control, Service-Oriented Architectures, Performance Validation and other management topics. To see more about the larger solution suite and to learn more about various products and solutions, look to the HP-Software website. You may also be familiar with Systems Insight Manager (SIM) and related tools, which focus on the HP-hardware ecosystems. All these tools monitor multiple systems and resources from a high-level, and are complemented by Glance and PA's ability to access OS kernel instrumentation directly to generate metrics for a "bottoms-up" drilldown to detail on a specific server. Glance and PA represent the difference between knowing "something is wrong" to being able to determine what exactly is causing a performance problem.

The ability to drill down into very detailed data on any given system is a key value of Glance and PA. The focus of this paper will be on Glance Pak for HP-UX, although much of the discussion can also be applied to the use of these products on their multivendor versions.

Product structure – "how did we get here?"

Glance is also called GlancePlus. In fact, you will see the name GlancePlus on most of the product literature, but it's all the same thing! Some marketeer, a very long time ago, thought that the name GlancePlus sounded better than Glance. You can use the terms GlancePlus and Glance interchangeably: I just choose to call it Glance.

The Performance Agent used to be called OpenView Performance Agent, and before that, it was named MeasureWare. The product names change, features are added and things are improved over time, but the basic functionality and use-cases remain the same.

Both Glance and PA have been available on HP-UX and other operating systems for many years, and are maintained via new revisions with improvements. Both products share some common Measurement filesets that connect to underlying kernel instrumentation to produce performance metrics. This measurement basis provides us with metrics as consistent as possible across different platforms and OS releases. The formal name for both Glance and PA together is the "HP GlancePlus Pak."

PA and Glance on HP-UX, individually or together as the Glance Pak, are delivered via Software Distributor bundles named by product number. The bundles point to SD-products that in turn point to the filesets that contain the files. Installing the Glance

bundle and the PA bundle separately will get you to the exact same result as installing the Glance Pak bundle. In other words: the Glance Pak truly is simply Glance and PA together.

Once you have the products installed on your system, you'll find the product files mostly under the directory /opt/perf, with executables in /opt/perf/bin. Some content in PA goes under /opt/OV directories. Since /opt/perf/bin is added to the system PATH after installation, I'll just refer to the executables henceforth without indicating the full path. The release notes are under /opt/perf/ReleaseNotes. Note that on the AIX platform, the /opt/perf directory structures are actually under /usr/lpp/perf, and the directory also varies on Windows PA, but it is /opt/perf for all other platforms (HP-UX, Linux, and Solaris). The configuration files that we'll be playing with, as well as other files created during and after installation, go under the /var/opt/perf directory. New default config files (like alarm definitions) are installed into the /opt/perf/newconfig directory, and at the end of installation they are conditionally copied into /var/opt/perf if there are not already config files there (from a previous installation). This prevents your customizations to the /var/opt/perf config files from being overridden when you update to a new release. If you want the new default configuration files installed, then remove them from /var/opt/perf before swinstalling the new version, or simply copy them over from newconfig after an update. You can also find some supplementary example files under /opt/perf/examples. I'll refer to a few of these examples below. In addition, we include electronic versions of the hardcopy manuals in directories under /opt/perf/paperdocs.

Some people are surprised to find out that they already have Glance Pak software installed and running on their HP-UX systems when they did not explicitly purchase it. Most often, this happens because you have one of the HP-UX Operating Environment software bundles that include the Glance Pak. It is also possible that somebody has installed trial versions, which are included as part of the HP-UX Application Release media. When you update to a new version, either update the whole Glance Pak bundle or make sure you update Glance and PA at about the same time: Glance and PA should always be kept on the same version on any given system.

Glance in brief – "don't blink"

Here's a quick overview of Glance: There are two user interfaces - *glance* and *xglance*. Both interfaces provide access to the same performance measurement data. The *xglance* interface is a X-window (Motif) Graphical User Interface that is easier to navigate but takes a bit longer to start up. The *glance* character-mode interface is useful when you don't have an X-display available. Note that if your primary display station is a PC, there are several X-window emulators available from different companies that allow you to display X-windows as well as terminal-based programs run from unix systems on your PC. I prefer to use character mode glance when I just want a quick look at a system, and I use xglance when I want to do more in-depth analysis. Because of the greater flexibility of the X-windows interface, xglance gives you access to some detailed metrics not available in character-mode glance.

A note about terminology: Just as there is no difference between using the terms Glance and GlancePlus for the name of the product, there is no difference between the using the names *xglance* and *gpm* when discussing the X-windows GUI. The xglance user interface used to be named gpm. On HP-UX, one name is just a link to the other in /opt/perf/bin, so there is really no difference.

When you start xglance, it will bring up the Main window along with any other windows that you had open the last time you ran it. In the Main window, you'll see information about the update intervals as well as summary graphs of some of the most important global performance data (CPU, memory, disk, and network activity). From there, you can use menu selections to bring up detail Report windows to show more information about various aspects of system performance. We'll go deeper into them later. Another button on the main window shows the status of any current performance alarms, which are configured using the Glance "adviser" functionality. The adviser is a useful way to access specific metrics and alarm on them using a script-like language (more on that below!).

You can adjust xglance's fonts, update interval, icons and other properties from the Main window menu. From this and any window, you also can get into xglance's online help facility. There's a lot of useful information contained in the online User's Guide. Finally, there's the "?" button which you'll see on every xglance window. That allows you to access the on-item help for the windows or metrics in the display. Click on the "?" button and then click on a metric or a window to see the online help. There are over a thousand different performance metrics available in Glance, any of which may be useful to you at some point. The Choose Metrics capability of every list window lets you customize which metrics are in the report, and you can define alternate sort columns and filters to highlight the most interesting data. To avoid getting lost in the complexity of all the different xglance reports and metrics, concentrate on the areas of performance most important to you, or to which the metrics displayed in the xglance Main window guides you. If you iconify xglance, then you'll see the CPU, memory, disk, and network activity continue to update live inside the xglance icon!

The character-mode glance display also shows the main areas of performance in the "bar graphs" at the top of the terminal emulation window. In the bottom of glance's initial screen is the Process List, which xglance puts into a separate window. You can use terminal function keys to navigate through character-mode glance's various report screens, or you can use shortcut letters to go between screens. If you type "?" inside glance, it will go to the Commands Menu screen which lists all the screens available. You can see what shortcut characters are available to navigate to any report screen. Note that many screens may have more than one page to them. They will say something like "Page 1 or 2" in the bottom right corner. To scroll to the next page, use the spacebar or the '+' key. To go back use the '-' key. when you scroll forward past the last screen it returns to the first. The "h" key will get you into the character- mode online help facility (which by necessity has less content than xglance's online help). A very useful topic in online help is the Current Screen Metrics topic which will show you the names ("monikers") for each metric.

From many of the glance reports, you can drill down to additional levels of detail. For example, when looking at the Process List you can use the 's' key to select a specific process. This takes you to the Process Detail report where all metrics shown relate specifically to the process id (pid) you selected. From there you can even get to MORE drilldown detail such as per-process memory regions, threads, and open files. Similarly, from a screen such as I/O By Disk, you can use the 'S' key (note use of capital S to select something other than a process in charmode glance) to drill down into further detail on a item in the list.

Although character-mode glance has access to all the same performance metrics as xglance, there is not enough space in the screens to display them all, but you can still access them all with the adviser functionality. Whichever user interface you choose, the way you use the Glance UIs are often the same: You take a quick look at the global performance indicators to see what general areas might be affecting performance, and then you drill down into that area of concern. At some point, you'll often end up looking at the Process List to focus in on specific activities that are at the heart of the issue you're trying to resolve. From a specific process perspective, you can drill further if needed into its open files, memory regions, wait states, and threads (for HP-UX and Linux). In general, Glance is the best tool in the industry for answering "what's going on right now."

Performance Agent in brief – "the names have been changed to confuse the innocent"

The Performance Agent (PA) product is what used to be called OpenView Performance Agent is what used to be called MeasureWare. It has been a multivendor systems performance agent since the early 1990s. While PA can be purchased separately, it is most often included in a bundle: either bundled with Glance as the Glance Pak, or bundled with the Operations Manager agent as the "Operations/Performance" product set. For OM/OVO-oriented people, PA is deployable from the OM console.

PA is structurally more complex than Glance, but it uses the same system performance metric instrumentation as Glance. PA's job is to log the performance data for future reference, use the metrics to trigger alarms, and pass alarm information as well as performance data up into datacomm daemons that connect with analysis products such as Operations Manager, Performance Manager and other reporting tools. The major part of PA is essentially a set of daemon processes that run in the background to collect, log, and process the performance metrics. Performance Agent is supported on all the same unix platforms that Glance runs on, and in addition it is supported on Windows servers.

At the heart of PA is scopeux (also known as "Scope"), which is the daemon that obtains the metrics. Scope uses the same measurement interfaces as glance and xglance do. Scope is collecting data continuously and logging process data every minute by default, and other (global, application, disk, network, and transaction) data every five minutes by

default. These intervals are configurable. Scope writes the performance metrics in logfiles that are read by the other product components. The `perfstat` script provides a good way to check on the daemon processes to make sure they are all up. For example, if you misconfigure an alarm definition in the alarmdef file, the perfalarm daemon will exit with an error and the perfstat script will highlight that perfalarm is no longer running. Most of the daemons write status messages to files under /var/opt/perf, and all the status files can be viewed using the –e or –t options to perfstat.

If you are not relying on other analysis products such as PM (or its precursor PerfView) to view your PA data, then you'll want to become familiar with the extract program that you can run to export data from the scope logfiles into different file formats. Using report files that specify to the extract program which metrics to export, you can customize the output to suit your needs.

Whereas Glance only runs when you invoke it, PA daemons are designed to run continuously in the background, starting up automatically at system boot time (controlled via the `/etc/rc.config.d/ovpa` file). The daemon start and stop script is `/opt/perf/bin/ovpa`. If you do nothing with the data, PA will continue to collect and log the data forever, but don't worry about running out of disk space because Scope will automatically delete old data from its logs once they reach a maximum size. PA complements Glance in that it's good at answering questions like "what went on earlier today?" and "has it been happening all this month?"

CPU Scenario: "Some hogs don't live on farms"

Let's look at a simple common system performance problem. Say your users are complaining about a server running slower than it should. You might telnet to and login to the system, and then run character-mode glance to see if it's obvious what the problem is. In this scenario, let's say glance's main screen shows the CPU Util bar filled up with activity. You would also see the current CPU percentage column at the right pegged at 100%. If you have the default process filters set up (the "o" screen in character-mode glance), the Process List will be sorted with the highest CPU user at the top. In this case, let's say you see the process "imahog" listed first, showing CPU utilization of 99%. Right from the Process List, you can see the username that is running the imahog process. If you then go into the Process Resource screen, you'll see lots more information about the process, including what kind of CPU time it is consuming. You get to this screen by typing "s" and then typing in the imahog process ID (pid), or by just typing ">". Additional process-specific metrics are available from there in the different drilldown screens. Some of this information can be very interesting to developers looking at how their programs are spending their time and what resources they are consuming, but in this simple example you're just trying to improve the system response time. At this point, you could use a `renice` command on the imahog process to change its scheduling priority. This allows other higher priority jobs to get more CPU time, or you could just kill it (Glance has shortcuts for these functions, assuming you logged in as superuser). Your best bet might be to check with the user who started the imahog process first!

If you had decided to use xglance instead of glance, you would have done the same unix login procedure and made sure the DISPLAY variable was set correctly in your shell. Once you started xglance, you'd see the global activity in the CPU Report window on your display, and requesting the Process List would have shown the imahog process chewing up lots of CPU. Manipulating xglance's Process List with its filtering and sorting capabilities can be fun, as there are many opportunities for customization. You could set it up so that only processes using more than 10% CPU or doing more than 10 disk I/Os per second are listed. You could highlight processes that are using more than 50% of any one CPU. You could sort by username to see what the different users are running. Another useful option in any of xglance's row/column textual reports like the Process List is that you can invoke the Choose Metrics screen to customize what fields are shown in the list. If your system is often subject to disk bottlenecks, you may want to pick up a metric like `PROC_IO_BYTE_RATE` to add to the Process List and then sort on it. If you are concerned about high system CPU utilization, you could choose to sort on System CPU %. If you are bedazzled by the sheer number of metrics and want to access the definitions, this is easy in xglance using the "?" button. Click on "?" and then click on a metric (for example, in the Choose Metrics window), and the metric definition will pop up. You'll notice some of the naming conventions (system-wide "global" metrics start with `GBL_`, process-specific metrics start with `PROC_`, etc). You can re-arrange the columns in xglance reports for easier viewing, too. Your changes will be automatically stored in binary-format user-specific and system-specific configuration files in your home directory (the shell command "`ls ~/.g??*`" will show them). This means that next time you run xglance on the same system you'll see all the customizations you had made in your last session. A quick way to return to the defaults is by removing those files.

What if the imahog process usually does useful work and you want to do a little research into when it started using excessive CPU (perhaps spinning)? This is where PA can come in handy. If you have Performance Manager (PM) available, it would be easy to connect to the target system and look at the history graphs, perhaps drilling down into the process data. All the data in PM comes from PA, unless PA is not present and you are getting the lightweight metrics from the Operations Manager agent. If you are not using a separate remote analysis tool like PM, then you can create a quick report template for PA's extract program to dump out the data. Try this: Copy the reptall file from /var/opt/perf into /tmp. Then edit the /tmp/reptall file, scrolling down to below the "`DATA TYPE GLOBAL`" line. Remove the asterisk from in front of the `DATE`, `TIME`, and `GBL_CPU_TOTAL_UTIL` metrics. Then scroll down in the file below to where you see the PROCESS metrics listed below the "`DATA TYPE PROCESS`" line. Remove the asterisks from in front of the `DATE`, `TIME`, `PROC_PROC_ID`, `PROC_PROC_NAME`, and `PROC_CPU_TOTAL_UTIL` metrics as well. Save the edited file and run the following command:

```
extract -xp -v -gp -r /tmp/reptall -b today
```

When finished, extract will have created two files in your current directory, `xfrdGLOBAL.asc` and `xfrdPROCESS.asc`, containing the metrics you requested. Both files will show data starting from midnight last night. In this example, the Global output might show you that the system was moderately busy starting at 07:00 but that CPU

percentage was near 100% starting at 13:00. Looking at the Process report output, you will see all the processes that PA considered “interesting” (more on that later). You can look for the imahog process records and perhaps see that at about 13:00 it started chewing up excessive amounts of CPU. Looking at what other processes were interesting around that same time period might help determine what caused imahog to spin… or not! Performance analysis, no matter how useful the tools, will always remain somewhat of an art.

This simple example touches on some of the capabilities of the tools. I like to keep sets of customized PA extract export report files around to help me with specific tasks. For example, I have a report file which concentrates on displaying CPU data as above, and another that reports on disk data. Information about report files for extract, extract options, and other details of PA can be found in the PA User's Manual.

By looking periodically at Glance and PA metrics on your important systems during times when they are running smoothly, you’ll get a good idea of what to expect. On busy systems, `GBL_CPU_TOTAL_UTIL` might not be very useful because it’s always near 100%. In that case, it might be good to watch `GBL_RUN_QUEUE` or `GBL_PRI_QUEUE` to monitor how many processes are typically waiting on CPU time. If the average number of processes waiting to run (blocked on Priority) gets abnormally high, users will experience poorer response time. To get more information specific to HP-UX performance analysis, I invite you to read a paper co-authored by a very experienced performance consultant: the HP-UX Performance Cookbook.

Let us say that you solved this particular problem, but you want to be more proactive about runaway processes on this system for next time. One option might be to keep a glance or xglance session active on your server so that you can keep an eye on the global metrics. Some administrators keep a bunch of xglance windows iconified on their workstation (I used to call this “Poor Man’s PerfView”)... the icons will blink red when an adviser bottleneck alarm goes off. PA serves this function even better, because its bottleneck alarms will be sent to Operations Manager (refer to the PA User’s Guide for detail). But, let's say you don't have OM and you want to set up a special alarm configuration inside PA to send yourself an email if processes start spinning again in the future. You could do the following: Copy the /var/opt/perf/alarmdef file into /tmp. Edit /tmp/alarmdef and append the following syntax to it:

```
hogpid = hogpid
PROCESS LOOP
{
  # Send mail to sysadmin when processes are hogging the CPU now
  # and have accumulated over 120 seconds (2 minutes) cpu time total.
  # Avoid processes with pids < 100 assuming they're system processes
  # and they know what they're doing.
  if (PROC_CPU_TOTAL_UTIL > 95) and
     (PROC_CPU_TOTAL_TIME_CUM > 120) and
     (PROC_PROC_ID > 100) and
     (PROC_PROC_ID != hogpid) then
  {
    exec "echo 'runaway process detected by PA' | mail me@mycompany"
```

```
    hogpid = PROC_PROC_ID
  }
}
```

Change the mail address from "me@mycompany" to something that makes sense for you, and then save the file. Run the PA utility program to verify its OK:

```
utility –xc /tmp/alarmdef
```

If no errors are found, copy it back into /var/opt/perf and issue the command:

```
ovpa restart alarms
```

This will tell PA's perfalarm daemon to reread `/var/opt/perf/alarmdef` and start processing the syntax that you just added. From then on, you should get an email notification of processes that may be stuck in loops. Here's another version of this same check, which has more bells and whistles in the exec action:

```
hogpid = hogpid
PROCESS LOOP
{
  # Send mail to sysadmin when processes are hogging the CPU now,
  # and have accumulated over 120 seconds (2 mins) of cpu time total.
  # Avoid processes with pids < 100 assuming they're system processes
  # and they know what they're doing.
  if (PROC_CPU_TOTAL_UTIL > 95) and
     (PROC_CPU_TOTAL_TIME_CUM > 120) and
     (PROC_PROC_ID > 100) and
     (PROC_PROC_ID != hogpid) then
  {
    exec "echo \"Possible runaway process detected by PA",
         "\\nname = ", PROC_PROC_NAME,
         "\\npid = ", PROC_PROC_ID,
         "\\ncpu util = ", PROC_CPU_TOTAL_UTIL,
         "\\nusername = ", PROC_USER_NAME,
         "\\ndetected on ", DATE, " at ", TIME,
         "\" | mailx -s \"runaway process alert from `hostname` \" ",
         "me@mycompany"
    hogpid = PROC_PROC_ID
  }
}
```

There may be several things here that aren't "intuitively obvious" if you haven't played with alarmdef syntax before. I typically just copy and edit from examples.

An email message from this syntax might look like this:

```
Possible runaway process detected by PA
name = imahog
pid =      11350
cpu util =  98.120
username = dgrumann
detected on 05/28/2008 at 18:02:00
```

There's a lot of flexibility in the alarmdef syntax, and it can be pretty tricky (especially complex EXEC statements like the one above), so take it easy, look at the examples under `/opt/perf/examples/ovpaconfig`, and always check your edited alarmdef

syntax with `utility -xc` before you move it into `/var/opt/perf/alarmdef`. The alarm syntax is explained in the PA User's Guide, and there are several examples of alarms commented out in the default alarmdef file itself.

In this example, you can use similar syntax inside Glance as well! If you didn't have PA available on that system, you could change the line:

```
"\\ndetected on ", DATE, " at ", TIME,
```

to be:

```
"\\ndetected at ", GBL_STATTIME,
```

then you could save just this alarm into a file and read it into xglance's adviser, or leaving it in a separate file on its own (say, `/tmp/procmail`) you could start up character-mode glance in the background like so:

```
glance -aos /tmp/procmail -j 60 &
```

Essentially, you're telling glance to run in the background and check once a minute for possible runaway processes. I call this trick "Poor Man's MeasureWare." The reason why you needed to change that one line when converting the syntax from PA alarmdef to Glance adviser is that the tools store time/date differently. Scope uses the metric names `DATE` and `TIME` while Glance uses `GBL_STATTIME`. There are only a few cases like this where the metric names in the tools aren't the same.

Symptoms – "Don't be alarmed"

Wading through all the performance data is important for some situations, but it can be tedious. The Glance adviser syntax and the PA alarmdef syntax are there to make things simpler, to add a bit of "intelligence" to the tools via generation of alarms based on botttleneck symptoms and other conditions. Both tools come pre-configured with a set of syntax that you can modify based on the system workload. The xglance online help has a section discussing the adviser syntax: you can get to it via the Help menu, select "User's Guide …" then "The Adviser" topic. For PA, the syntax is very similar and is discussed in the PA User's Guide's "Performance Alarms" chapter.

For character-mode glance, the default adviser syntax is in the `/var/opt/perf/adviser.syntax` file. The first time you install Glance, this file is created, but it won't be overlaid if you re-install so your previous changes are preserved. If you get a new version of Glance, you can compare the latest default adviser.syntax file in /opt/perf/newconfig to the one in /var/opt/perf to see if you want to pick up any changes. With xglance, its adviser syntax is encoded in the startup file in your home directory that also holds your window customizations. You can reset it to the default via menu selections in the xglance Adviser windows. Both interfaces have the same default alarms. Look at the files, and you'll see they're divided up into sections. The "symptoms" are definitions that we've come up with to represent probabilities for bottlenecks. For example, the CPU_Bottleneck symptom assigns percentage probabilities based on metric values for some CPU metrics. It looks like this:

```
symptom CPU_Bottleneck type=CPU
rule GBL_CPU_TOTAL_UTIL        >   75  prob 25
```

```
rule GBL_CPU_TOTAL_UTIL        >   85  prob 25
rule GBL_CPU_TOTAL_UTIL        >   90  prob 25
rule GBL_PRI_QUEUE             >    3  prob 25
```

This means that if the average CPU utilization on the system exceeds 75%, the probability of a CPU bottleneck is 25%. If CPU utilization exceeds 85%, then the probability goes up to 50% (the "probability" fields at the right are added up for all the rules which evaluate true). If CPU utilization exceeds 90%, the probability of a bottleneck goes up to 75%, and then things depend on the Priority Queue metric which is the average number of processes waiting for CPU time.

Consider this: If your CPU is 100% busy, meaning that processes are using all available CPU resources, this is a good thing! You want the money your company invested in the processing power of the machine to be used. There is the potential of a problem only if the CPU resource is saturated *and* processes are kept waiting for the CPU. If the CPU is busy, and more than 3 processes are, on average, waiting for the CPU, then Glance has decided there is a 100% chance of a CPU bottleneck occurring.

These rules are based on experience from many systems, but don't apply perfectly in every case. On very active servers, the CPU may always be busy and the Pri Queue may often run high even when things are going smoothly. You don't want your xglance icon flashing red all the time, so on some systems you may want to tweak the rules so that alarms are more likely to go off only when response time or throughput is suffering. Remember that what you want to improve is overall application performance. These performance metrics are indirect measures of that. Direct application response time metrics are harder to come by (though we'll get to that later).

The Glance symptom statements for the other bottleneck areas are similar to the one for CPU. They combine various metrics that are good indicators for a resource into a bottleneck probability, which is then reflected in alarms and alerts. In Glance, we also pre-define some alarms based on system table utilizations. Something like a system running out of available swap space is not so much a performance problem as a configuration issue, but it is included.

PA similarly has bottleneck symptoms and alarms defined in its default alarmdef file. Like adviser.syntax, the PA alarmdef lives in /var/opt/perf and is not overwritten when you re-install so that your changes are preserved across updates. The default alarmdef will be in /opt/perf/newconfig, and there are several fancier examples under the /opt/perf/examples/adviser and /opt/perf/examples/ovpaconfig directories. Check them out!

PA has a facility built into the utility program that makes tuning your alarmdef file easier. If you run the command:

```
utility –xa –D –b today-1
```

it will process the historical scope logfile data and show you how your current alarmdef file would behave for the last two days. Leaving off the `–b today-1` parameter lets you go back further, but it may take longer to process. See "`man utility`" or the PA User's

Manual for a description of its options. Alarms are shown and EXEC statements are printed (but not actually executed).You'll probably see different alarms start, repeat, and (hopefully) end over time. You can look at the historical data with extract or one of the analysis tools to see why alarms went off at different times. If alarms are never going off and your users never complain about response time, then sit back and read your horoscope instead of this article! If alarms are going off too frequently, or they go off at odd times when you know the system is behaving, or (worse) they aren't going off during periods when you know the system is in trouble, then it's time to review the thresholds and alarms and customize them better so that next time they'll alert you when performance is an issue.

A lot of people have complained to me about the default alarmdef bottleneck alarms going off too frequently on their systems. I have heard stories of people deleting all the alarms from alarmdef, just because the default thresholds were too low for their workloads. But: not generating alarms at all may be worse than generating too many alarms! If you have this problem, I advise you to try editing the Symptom thresholds (in other words, edit alarmdef to make it harder for symptoms to trigger alarms). Use utility's –xa function to see how your changes would have affected historical alarming. As an example, here is a replacement Network Bottleneck symptom that I created which tends to be more useful than the current default on busier systems:

```
symptom Network_Bottleneck
rule GBL_NET_UTIL_PEAK     >    0 prob GBL_NET_UTIL_PEAK
rule GBL_NET_COLLISION_PCT >   20  prob 30
rule GBL_NET_PACKET_RATE   > 9000  prob 15
rule GBL_NET_OUTQUEUE      >    1  prob 15
```

Metrics – "No answers without data"

So what are the "best" metrics to look at to solve performance problems? The answer is: "It Depends!". This is the first rule to learn about the art of system performance tuning. Every system's configuration and workload is different, and every problem might require you to look at different metrics to characterize what is going on. Nobody can give you a short list of golden metrics that will characterize all performance issues. With experience on your servers, you will find that some metrics are more useful than others.

When looking at a system's performance, it's best to begin at the global level. Glance and PA's bottleneck alarms and their global metrics give a summary of how the system as a whole is behaving. The global metrics shown in Glance's main window and the ones used in the bottleneck symptom definitions are important indicators. From the global level, you drill down into different metric classes. You can view disk I/O activity from the device perspective, and see filesystem space utilizations as well. There's a metric class that gives data for each individual process. There's a class of metrics to track information about each network interface. The sum of input packet rates (`BYNETIF_IN_PACKET_RATE`) for all network interfaces on the system equals the global metric `GBL_NET_IN_PACKET_RATE`, just as the sum of physical I/O rates

(BYDSK_PHYS_IO_RATE) for all disks equals the global metric GBL_DISK_PHYS_IO_RATE. In turn, some of these class metrics are broken down further. For example, the Disk I/O rates are a sum of the Read and Write rates.

Many metrics are subsets or recalculations of others. The global metric GBL_CPU_TOTAL_UTIL is the average utilization of all the CPU resource on the system over the collection interval, while GBL_CPU_TOTAL_TIME is the amount of CPU busy time over the last collection interval. The multiple metrics are there so you can use whichever you feel most comfortable with. For example, if you are benchmarking a program you may care more about how many CPU-seconds it used than about how busy the CPU was while the program was running.

In the default alarmdef file, down where there are some commented-out examples, you'll see a few references to APP_ metrics. These are application metrics. This is a special set of metrics that summarize data from groups of processes to make it easier for you to understand the performance data. For example, if there are 3 processes running in the "backup" application and each of them is doing 100 I/Os per second, then APP_DISK_PHYS_IO_RATE for the backup application would be 300. Applications are defined in the /var/opt/perf/parm file, which is a file shared by both Glance and PA. This is another file that isn't overwritten during software update, in order to preserve your changes. The default is /opt/perf/newconfig/parm. We ship some application definition defaults that are just examples. I suggest that you change them, if you plan on using the APP_ metrics, because the applications running on every system are different! The PA Installation and User's Guide go into some detail about all the contents of the parm file, but the important thing to understand about application data is that you don't *need* to define applications: PA and Glance work perfectly fine if there are no application definitions in the parm file at all! Every process that doesn't match an application definition gets its data bucketed in the "Other" application. Application definitions *only* affect the APP_ metrics... there is no affect on the GBL_ or PROC_ metrics. If you want to look at the overall performance characteristics of an application conveniently, then you can set up your parm file to match your environment and use the Application metrics reported in both Glance and PA. Examples of some application definitions you could use are in the /opt/perf/examples/ovpaconfig/parm_apps file.

Another important thing to remember about application and process data is that the application data will reflect activity for all the processes that are grouped into the application, whether or not any of those specific processes were actually reported. Glance has filters that allow you to restrict the number of processes shown in the Process List, but even the processes not shown in the Process List still contribute to the application and global summary data. In other words, filters you set up in one window will not affect data in the other windows.

PA also has "interesting" process filtering, which controls what processes are logged by Scope. The process thresholds for PA are set in the parm file. The default process threshold line from parm looks like this:

```
procthreshold cpu = 10, memory = 900, disk = 5, nonew, nokilled
```

This means that only processes using more than 10% of any CPU, or having greater than 900 megabytes virtual memory set size, or generating over 5 physical disk I/Os per second average during the collection interval for process data will be defined at “interesting” for Scope, and thus logged in the PA logfiles. If there were 100 processes in an application, each of which only used 1% of the CPU, then none of them would be logged individually in the process data class with the default thresholds. However, the application and global metrics would still reflect the activity. Its wise to tune the filters for process data being logged by PA by adjusting the parm thresholds to suit your needs. If you want to see a lot of historical process-level detail, you can log processes using more than, say, 1% CPU over an interval instead of 10%. Be cautious, though: I more often see "too much" process data being logged by Scope as opposed to too little. If your parm thresholds cause a lot of processes to be logged, it will increase the scopeux daemon’s overhead, especially on busy systems, and it might cause the process logfile (`/var/opt/perf/datafiles/logproc`) to fill up and roll over more quickly than you would want. However, such settings may sometimes be useful for debugging problems. By controlling the amount of data being logged, the frequency of logging, as well as the maximum size of the logfiles (all defined in the parm file), you can keep the right amount of historical data to meet your needs. If you rarely look at the PA process data, then set your process logging thresholds higher. The other data classes (and Glance) will not be affected.

Another thing to keep in mind when working with the performance metrics: xglance can "get at" every system performance metric in its row/column reports. Look for the "Choose Metrics" under the "Configure" menu of any xglance row/column type reports, and use it to browse all available metrics and perhaps add them to your reports.. you may want to scroll or rearrange columns to make your favorite metrics more visible. Likewise, this is often the best way to get at online help explaining any of the perf metrics. For example, useful metrics like Network Interface byte rates (`BYNETIF_IN_BYTE_RATE` and `BYNETIF_OUT_BYTE_RATE`) are available in xglance’s Network By Interface report if you choose them. Glance character mode displays a subset of the xglance metrics because of its screen limitations. In either case, the adviser functionality can access the full set. So, for example, you can write new default bottleneck alarms for glance or xglance that use `BYNETIF_*_BYTE_RATE` even if they're not shown in the screens. Scopeux logs a subset of the metrics that xglance has available. The list of PA metrics is in the `/opt/perf/paperdocs/ovpa/C/methpux.txt` file. Any of those logged metrics can be used in the PA alarmdef symptom and alarm definitions, and those same metrics are what appear in Performance Manager and other analysis tools.

Memory Scenario – “There’s a hole in my bucket”

Let’s go through another common problem situation. Let’s say that you’re managing a server with a mix of applications. You’ve recently installed a new version of a client/server application that your development group gave you, and although online users reported good response times at first, they started complaining about the application being slow after a couple of days, and eventually the server process “ileek” aborted and

had to be restarted. Now it's a few days later and users are complaining again that it's slower.

Since you know history is repeating itself, you might want to take a look at the historical data from PA. What was going on with the system when the application response degraded last time? Let's say that there was a noticeable sustained increase in disk activity (`GBL_DISK_PHYS_IO_RATE` and `GBL_DISK_UTIL_PEAK`) until the time the application was restarted. You could look at the type of disk I/Os going on, and which disk and logical volume was busy. Sometimes by knowing what's on the disk (datasets, home directories, swap) you'll have a clue as to what was happening. You can also look at the PA application and/or process data to find anything out of the ordinary. Comparing the data for the specific client/server application over time, you might see that the Virtual Memory (VM) I/O rate (`APP_DISK_VM_IO_RATE`) and the Virtual Set Size summation (`APP_MEM_VIRT`) was increasing. In the process data, you might see that the process "ileek" had more memory faults (`PROC_MAJOR_FAULT`), VM I/Os, and a steadily increasing VSS (`PROC_MEM_VIRT`).

These are classic symptoms of a memory leak in the program. Your new version of ileek is probably repeatedly allocating memory and not releasing it. You could look at the latest logged PA data to see if that same trend is occurring now. In general, this type of problem will often first show up as a decrease in the amount of system free memory (`GBL_MEM_UTIL` will climb to 100%) along with a sustained increase in the amount of swap space reserved (`GBL_SWAP_SPACE_UTIL`). As memory pressure increases you'll start seeing more VM I/O activity (`GBL_MEM_PAGEOUT_RATE` is an excellent indicator) and you'll start seeing swap space or memory bottleneck alarms from the Glance and PA. It's usually pretty easy to spot the application and process that is causing the problem by browsing the data over long periods (if a process leaks 100 kilobytes a minute, its VSS will increase over 100 megabytes a day).

Glance can come in useful at this point to get more detail. Just as in the PA data, you'll see the same set of metrics having larger than "normal" values for your system. The Glance adviser may tell you that there's a memory bottleneck or a disk bottleneck or both (as HP-UX is doing a lot of VM I/O to the swap devices). In the xglance Process List, you can set your sort fields to the Virtual Memory field, and the biggest memory users will pop to the top. Select the ileek process and bring up the Process Memory Regions report window. You can browse all the memory regions that the process has allocated, looking for the one that has a very large VSS. Often this will be Type=DATA, meaning it's the process's own heap space. At this point, you call in the developer who gave you the new version of the application and bawl them out.

So we solved this memory hog issue this time. You may want to draw up some PA alarmdef syntax to watch for this in the future. If possible, I'd recommend using Application data instead of Process data for PA alarms. This presumes that you've made the time to edit the parm file applications such that you are capturing what you want. For our example, let's presume you've done this for the DBServer application. The parm file entry for it might look something like this:

```
      application = DBServer
      user = dbadmin
      or
      file = ileek,idbmaint,idbdaemon
```

Remember that when you change the parm file, you need to restart Glance or PA ("ovpa restart") to pick up the changes. In order to set a good value for the VSS threshold in our new alarm, you can use Glance and look at the Application List's Virt Mem (APP_MEM_VIRT) field for DBServer to see what a "normal" value is. The application metric is the sum of all the processes in that application, so for example if ileek and idbdaemon share access to a large shared memory segment, its VSS contribution would be reflected twice in the application's VSS metric. Let's say you find that 80 megabytes (about 80000kb) is a normal value for this application on your system. Our alarm could be set to send email if we see it go over 100 megabytes. We can replace our alarmdef memory hog syntax with this:

```
# Watch for DBServer application using over 100MB memory VSS sum:
VSSthreshold = 100000
application loop {
  if (APP_NAME == "DBServer") and
     (APP_MEM_VIRT > VSSthreshold) then
  {
    exec "echo 'DBServer app memory VSS alert' | mail me@mycompany"
  }
}
```

Although this PA syntax example works well, when it starts going off you'll get email messages continuously until you do something to bring the VSS for the DBServer application back down. This might be inconvenient, and if the condition starts going off at night then you could get a pretty full mailbox by morning. This is where the PA alarmdef ALARM syntax statement comes in useful. You'll notice in the default alarmdef file that the bottleneck alerts are used inside specific alarm statements (as opposed to loops or "if" statements). Alarms do a couple of nice things for you. They allow you to define a duration over which a condition is true before it starts, and a repeat interval (if desired) to remind you the condition is still true. An alarm can have a different action specified for when it starts, repeats, and ends. Finally, alarms let you reference specific parm file applications without needing to use an application loop (in fact, you can't use the loop construct inside an alarm statement). Here's a version of our memory hog syntax using an alarm:

```
# Watch for DBServer application using over 100MB memory VSS sum:
VSSthreshold = 100000
alarm DBServer:APP_MEM_VIRT > VSSthreshold for 5 minutes
  start {
    yellow alert "DBServer app memory threshold exceeded"
    exec "echo 'DBServer app memory alert' | mail me@mycompany"
  }
  repeat every 60 minutes {
    yellow alert "DBServer application still hogging memory"
    exec "echo 'DBServer app alert continuing' | mail me@mycompany"
```

```
}
end
  reset alert "DBServer memory demand now below threshold"
```

It's best to try to set up most of your alerts inside alarms like this so you can control the frequency of notifications. Alerts will automatically generate Operations Manager events if you have also installed OM Agents on your systems. There are examples of cpuhog and memory hog alarms in `/opt/perf/examples/ovpaconfig/alarmdef_procloop` that show how you can create alarms on data from process loops.

If you want to tune up your application definitions in the parm file, try using the xglance Application List. Double clicking on any app in the list will bring up the list of processes that are being bucketed in that application. This is very useful especially if the "Other" application shows a lot of CPU consumed. You can change the parm file to get processes bucketed better, though you'll need to stop and restart xglance to see those changes reflected in its Application List. When I do this, I usually delete applications (like some of the ones in the default parm) that don't make sense on the particular system I'm working on, and then add ones which logically group the processes according to my knowledge of the workload. To make application data useful, you will want few processes that are using a lot of CPU bucketed under the "Other" application. Once the applications make sense for you as shown in xglance, you can do a "`ovpa restart`" so that the PA Scope data will reflect your changes. Looking back at the historical PA process data later, you can look at the `PROC_APP_ID` to see whether you need to make more changes (Application Ids are assigned according to the order of your application definitions in the parm file).

Multiprocessor considerations – "Give it your 200%"

When looking at CPU metrics on multiprocessor systems, you should be aware that the application and global CPU utilization metrics are "normalized" in Glance and PA. That means that these classes use the number of active processors to help calculate relative CPU percentages. For example, imagine a 3-way MP system that has two processes in the same application, both looping. They could *each* use nearly 100% of a processor (CPU). Over a 10-second interval, each process uses nearly 10 seconds of CPU time, so the application and the system as a whole have used nearly 20 seconds of CPU time in 10 seconds of elapsed time. When calculating utilization from these numbers, we leave the process data alone (i.e.: each process would show that it used nearly 100% CPU over the interval) but we divide application and global CPU activity by the number of active (non-disabled) processors. In this case, both the application and the global total CPU Utilization would be about 66%, instead of 200%. When you are looking at the overall system workload, or comparing one system to another, you usually consider CPU utilization as a number relative to the total processing power of the system. So, 66% global CPU utilization means that 2/3^{rds} of the entire processing power of the system was used over the interval.

This explains why you can see, for example, processes listed in the Glance Process List or PA Process data for which the CPU percentages in an interval, if you sum them up, exceed 100%. Things get even more complex with multi-threaded processes, there each thread of execution can consume CPU independently of the others. You could imagine a process on a 4-way MP system that has three threads all of which are looping. In this case, the process as a whole is using 30 seconds of CPU time over a 10-second interval. Both PA and Glance would report the process using 300% CPU. The application and global metric classes however, would normalize this CPU utilization and report the overall CPU resource as 75% busy.

Virtualized Systems – "What's real?"

Generally Glance and PA concentrate more on local system resources, processes, and applications than on virtualization. Higher-level tools that are not system-centric can provide perspective on multiple systems which may be attached to shared hardware via one of several virtualization technologies. Every partition of a server (or, every "guest"), is a separate logical system, and the standard Operating System instrumentation interfaces are generally unaware of a larger virtual environment. As an example, if a partition or guest is configured to use a maximum of 2 CPUs on a physical system that has 16 CPUs, then the metric `GBL_NUM_CPU` (maximum number of CPUs) in Glance and PA will be 2. The `GBL_ACTIVE_CPU` metric will vary depending on how many processors are currently enabled in the partition, which can in some cases change dynamically. Likewise, Global memory metrics will reflect the amounts assigned to the partition that Glance and PA are running in. Processes shown in the tools are the ones running on the image that Glance or PA are running on. So, as a general rule, running the Glance Pak inside a compartmentalized guest or partition gives you a view of system performance relative to the logical (virtual) bounds of that partition.

Now, in certain cases there is additional instrumentation available specific to some virtualization technologies (such as, for example, HP Integrity Virtual Machines) that allows Glance and PA to "see" more of the virtualization context. In these cases we have additional metrics which can show, from the guest perspective, some of the underlying physical metrics related to it. Also, there can be additional metrics available on the server which give data on a per-guest basis. This is a topic is discussed at length in a Virtual Performance Whitepaper that I wrote, as well as in the product documentation.

Managing collection – "Update Thyself"

Some people who have used PA for a long time may not know about some of the newer configurability features. The new options are all described in the default parm file (/opt/perf/newconfig/parm), as well as in greater depth in the PA User's Guide's chapter dedicated to the parm file. The PA User's Guide is on the HP-Software product manuals site. You can now control the amount of data logged in Scope's logfiles not only by the size of the logs but, alternatively, by the number of days minimum that you want to keep.

You can also now configure the Scope logging intervals for the Process class and the other data classes via `collectioninterval`. The Process logging interval can now be as frequent as every 5 seconds and the other classes as frequent as 15 seconds.. but *be careful* when logging frequently, as the overhead of Scope will naturally increase and your logfiles will fill faster. Also added to PA is more flexibility in all the data classes over choosing when to log specific instances based on activity. For example, there is no sense logging every single disk device every interval when activity may be low. You can choose to only log disks, for example, if they were over 10% busy during the interval (`disktheshold util = 10`). Setting good thresholds will make sure more of your logfile space is consumed with “useful” records instead of information on idle disks, filesystems, and network interfaces.

Another recent feature in PA is that you can turn on full process command logging with the `proccmd` setting, but of course the more that is collected the more space is consumed. For the process command strings, Scope is smart enough to only keep one copy of each unique command, therefore processes that are logged over and over do not waste space, but if you are on a busy system with long command strings that also change frequently (sometimes seen in java environments), the new process command logfile(s) `/var/opt/perf/datafiles/logpcmd*` may grow large.

PA has only recently added flexibility to its collection intervals, but Glance has always had flexible update intervals. The default update interval in charmode glance is 5 seconds but you can change that with the 'j' key. The default xglance update interval is 15 seconds, changeable via the Configure->Measurement menu. One trick I like to use is to set the update interval really high (like, say, 300 seconds), and then just request updates when I want to see them via the Enter key in charmode glance, and via control-U in xglance. This way, I do not need to hurry as I switch between screens showing the same time range.

I often hear a concern from sysadmins about "missing data" from in-between update intervals (collection intervals). The idea is that frequent update intervals must be required in order for a tool to capture short-lived system activity. This is not true of Glance and PA especially on HP-UX: while more frequent sampling intervals do provide finer granularity, you are not *missing* the activity that occurs between updates, because our measurement facility is active continuously! As an example, let's say you had a 1-minute collection interval, and a process started just after the last update and completed before the next update (running for less than a minute total): one might think the process would not show up, because it was not active during any collection sample, but it does! This is because the underlying instrumentation has recorded it. This is why you see "died" processes show up in Glance and logged by Scope in PA. The processes had terminated when the collection interval occurred, but we still captured their activity and their effect on the system. If you are using PA on *non*-HP-UX systems, you may want to look up the `subprocinterval` parm-file setting whch can adjust scope measurement sampling, where the OS instrumentation is not as robust.

I encourage people to use "sane" collection intervals and thresholds, meaning ones that do not produce more data than is reasonable to be looking at on an ongoing basis. You can always decide to "ramp up" collection for short periods of time by lowering the update interval (and/or the logging thresholds, for PA), then returning it perhaps back to the original parm settings for the long-term. When you lower the update interval (thus increasing the frequency of collection), you not only use up logfile space faster but you also correspondingly increase the overhead of the scopeux collection daemon, and also increment datacomm (coda) and alarming (perfalarm) cpu demands.

To see space statistics about the Scope logfiles, run the command "`utility -xs -D`" and pay particularly close attention to the summary statistics at the end, where you can see in the Process Summary Report which logging thresholds are relevant for processes, and the Megabytes logged Each Full Day, for Process and the other data classes. If you are logging many many MB of Process data every day, and perhaps not able to keep as much history as you would like (because of rollover), then you can increase the size of the logproc file via the size parameter, or bump the procthresholds up in the parm file to reduce the amount being logged. You can also create a parm-file application grouping specifically for the PA processes (the ones that show up in perfstat output) to monitor collection overhead on an ongoing basis. Measuring the overhead of any management function is a good general policy to follow (especially for managers :-).

When Scope rolls older data out of the logfile, it will be lost unless you choose to archive it somehow. If you have Performance Insight, its data warehouse functions essentially archive the PA data that it gathers – in the PI database. In that case you may not be concerned about how much data you are keeping on the monitored system. Independently from PI, you can still of course back up PA's logs in any number of ways, though it is a good idea to shut down Scope temporarily (`ovpa stop`) while you are archiving. Archiving logs are another topic of the PA User's Manual, and some people do have extract processes to keep data for long periods, while others don't try to extract but instead just back up the raw Scope logs or copy them to a central system.

If you move Scope logs from one system to another, be aware that you can access logs originating from another system in a "proxy" fashion with tools like PM. For example, if you copied the Scope logfiles originally from SystemA to SystemB under, say, /tmp, then you can make those SystemA logs accessible to PM by: 1) Adding an entry like "`DATASOURCE=SystemA LOGFILE=/tmp/logglob`" to the `/var/opt/OV/conf/perf/datasources` file on SystemB, then 2) Restarting PA on SystemB to pick up the new datasource, then 3) in PM, add a node named `[SystemB]SystemA`. This tells PM to connect to the SystemB node but instead of looking at the default SCOPE datasource, look for the SystemA datasource instead. Remember when graphing from copied logs that these logs will have older (non-current) data in them, so you will need to specify PM graph date ranges explicitly, or set the date defaults to graph ending "Last" instead of "Now." This is how performance consultants may copy your logfiles back to their own systems for remote analysis. This proxy trick may also be useful to you in case you have a lot of people analyzing archival logfiles and you want to

do it “offline” by hosting those archive logs on a management server instead of the production systems that created them.

Importing data into PA – “Log whatever”

A feature of the PA product that I will not discuss in detail is its Data Source Integration (DSI) capability, which is a way to put metrics into PA for access by higher level tools alongside the system performance data that it collects automatically. This is a complex topic, and the Data Source Integration manual distributed with the PA product goes into a lot of detail with examples of how to import data from other tools to log and alarm on. Smart Plug In products use this capability to, for example, log database application information into PA. It is not common practice, but you can use this facility to set up data feeds from your own applications or from other monitoring tools. As an example, you might find a metric in Glance that is important to you but does not happen to be logged by Scope and you could write a script that pipes glance adviser output into a DSI datafeed. In this way, any metric available in Glance can be imported into PA.

Here is an overview of the basic steps to create a DSI datafeed for PA: Once you figure out what metrics you will want to log, you create a spec file using DSI's syntax. Then, you initialize the DSI log using the spec file and the `sdlcomp` program. Insert some data into the logfile using either your own data feed or sample data from `sdlgendata`. Whatever the method, you pipe the data as stdin into the `dsilog` program, pointing at the DSI log you created. You can verify that the DSI log has your data in it be using the `sdlexpt` program. At that point, you can reconfigure PA's datacomm to recognize this new data source by adding an entry for the log to the `datasources` (or `perflbd.rc`) file. Restarting PA will pick up the new data source, at which point PM and other PA clients can access and graph the data. Details are in the DSI manual.

Another custom feature of Glance and PA is the Application Response Measurement facility. This is a Software Development Kit that contains an industry-standard ARM Application Programming Interface (API). Our API is version-2 of ARM, which is a C-language level interface developers can code to, then link their created applications to the ARM library in order to instrument transactions within their programs. The result is that response time and throughput metrics will be generated from the activity of their applications. This is the “transaction” data class (`TT_` metrics) that you see in the tools. The Tracking Your Transaction manual (available online under /opt/perf/paperdocs/arm/C) discusses this facility of the performance tools in depth, and touches on the subject of defining Service Level Objectives for your applications. If you have an internal software development group, you may want to make them aware of this technology. These days, I personally find that fewer organizations are writing new code in-house, and so interest in coding to the ARM C-API has waned. Also, there are other industry tools (including other HP-Software products such as RUM and BAC-Diag) which can analyze specific applications without programming changes.

If you do have ARMed applications running in your environment, you can establish Service Level Objectives (SLOs), which the tools can use to monitor and alarm. To make good use of the ARM data you'll want to spend some time tuning the `/var/opt/perf/ttd.conf` file so that the response time bins and SLO thresholds make sense for your system. The most important transaction metrics are the number of completed transactions (`TT_COUNT`) and the number of completed transactions which exceeded your SLO threshold (`TT_SLO_COUNT`). You can also monitor the distribution of completed transactions in the response time bins to get a relative feel for what percentage of transactions are completing quickly or slowly and comparing this to the system-level metrics. By keeping watch on how many transactions exceed your service level objective, you can be more proactive about monitoring application response and responding to problems. We ship a sample alarm that is based on SLO violations in the PA default alarmdef, and there's some example adviser syntax shipped with Glance in the `/opt/perf/examples/adviser/arm` file.

Exporting data from the tools – "Feed the Need"

As mentioned before, there are Reporting tools available such as Performance Manager, Reporter (part of Operations Manager for Windows), and Performance Insight that "feed" off PA data. They connect to systems running PA with (proprietary) datacomm protocols, and bring the metrics to a local system. PM just brings metrics on an ad-hoc (per-request) basis, whereas Reporter and Perf Insight schedule daily "gathers" from agents to populate centralized databases. Once gathered to their central database, you can use those tools to customize reports and otherwise make use of the PA data. However, I am often asked about how to gain access to the system performance metrics more directly.

If you want to pump performance metrics into another process or program, I suggest one of three methods: 1) using the glance adviser-only functionality, 2) using the data output from PA's extract –xp command, or 3) export from a central system running Performance Manager, using the ovpmbatch facility to access PA data from any system PM can access. There is no way to access the raw Measurement data from directly programmatically, the raw data source of Glance and PA: There is no API (link-library) for performance metrics independent from the OS itself.

A quick example of using glance's adviser-only functionality:

```
glance -aos /opt/perf/examples/adviser/activity
```

A quick example of using extract's export functionality, to create ASCII export files of global and process data since yesterday:

```
extract -xp -b today-1 -gp -r /var/opt/perf/reptfile
```

A quick example of using PM's ovpmbatch functionality to dump a few metrics from a remote system (replace mysystemname):

```
ovpmbatch systemname=mysystemname graphtype=tsv class=GLOBAL \
  metric=GBL_CPU_TOTAL_UTIL metric=GBL_RUN_QUEUE \
  daterange=2hours enddate=last
```

You could use these examples and modify them to meet your needs.

Various other products specific to capacity planning, such as SAS, HyPerformix, or HP Capacity Adviser, use one of more of these methods to pull metrics from PA into their models.

Miscellaneous Tips & Tricks - "impress your friends"

Right-clicking (mouse button3) on any metric in xglance will bring up that metric's online help text. This is a shortcut alternative to using xglance's "?" button.

Double-clicking (button1) on a process in xglance's Process List will bring up the Process Resource report, which shows the process' full command line among other interesting metrics. You can also go from there to other process-centric reports such as Process Open Files, Memory regions, System Calls and Thread List.

Pleasant color scheme for xglance, try the command:

```
xglance –fg white –bg "#404040"
```

You can also change app-default settings in /`var/opt/perf/app-defaults/$LANG/Gpm`.

For a nice big charmode glance display, try:

```
nohup hpterm -fg white -bg black -fn iso1.20b -exec glance
```

To drill to a particular process name in xglance not visible in the Process List: use Configure->Filters menu, click in the "Disable All Filters" box in the upper-right, click "OK" to close Filters screen, back in Process List use Search->Find and type in desired process. It will highlight the process, then hit Enter to bring up details screen.

If you add the full command line to xglance's process list (via Choose Metrics and clicking on `PROC_PROC_CMD`, clicking OK and scrolling over to the right), you will notice that by hovering the mouse pointer over a process who's command line does not fit into the column, a "tooltip" will pop up that displays the full metric value. The xglance tooltip popups will come up any time a field in a row/column display is truncated, by default.

Character mode glance stores its threshold settings in a (binary format) `$HOME/.glancerc*` file.

X-window mode xglance stores its threshold, sort, highlight, filter, and other GUI settings in a (binary format) `$HOME/.gpm*` file.

The only Glance screen affected by the parm file is the Application List. Nothing in parm has any effect on any other class of data shown by Glance besides the `APP_` metrics shown in the Application List. If you want to have a special set of application definitions for your own use, then you can create a file in your home directory named .parm that has the application definitions you want to use. If Glance sees a `$HOME/.parm` file when it starts up, it will use that and ignore the `/var/opt/perf/parm` file.

The coda daemon is used purely for datacomm in PA, even though it is part of filesets that are shared with the Operations Manager agents. When the OM Agent (OVO Agent) is also installed, then coda performs datacomm *and* does lightweight metric collection for OM.

The "old" DCE/RPC datacomm daemons perflbd and rep_server will only be running on your system if you have updated from an old (3.x version) PA. If you want to run them, for example to connect to the old PerfView product, you can edit the startup config file `/etc/rc.config.d/ovpa` and change the `MWA_PROTOCOL` setting from `http` to `both`.

If the `perfstat` script tells you that some processes are not running, use `perfstat –t` to check the ends of the daemon status files, and perhaps issue a "`ovpa restart`". If all the daemons still do not come up, there may be a problem you should call HP-Support about.

A good way to check Scope logs is "`utility –xs –D`". A good way to check PA datacomm locally is "`utility –xa –D`". A good way to check datacomm remotely is "`/opt/OV/bin/ovcodautil -ping –n` *`systemname`*".

In addition to the manuals for PA, there are man-pages for extract, utility, perfstat, arm, midaemon, and ttd. Most programs also have usage strings (for example, "`extract -?`" to see its options).

If you are using Ignite-UX or some other method to clone systems, be sure to "`rm /var/opt/perf/datafiles/log*`" prior to creating your image. This will start PA logging fresh on the new (cloned) systems and not carry history from the originating system. You do not want your Scope logs on different systems to appear to all originally come from the same system.

If you want to copy PA logs from one system to another, as an example for remote analysis, then you can just copy `/var/opt/perf/datafiles/log*` , however I prefer to use the `–z` option of perfstat which allows you to create a tar archive of all the logs and config files and status files, all of which may come in handy. The `perfstat –z` command will produce a (large) `/var/tmp/perfstat.tar.Z` file that you can move to another system, then use `uncompress` and `tar –xv` to unpack. Be sure to do this in a directory other than /var/opt/perf, or you will overwrite the live PA data with the archive!

Conclusion – "where to go next"

There’s a lot of information packed into this paper, but there is also a lot more to the Glance Pak that I haven't gone into. Rather than try to understand all the capabilities and extensions to the products right away, start with the simpler task of monitoring your workload to gain an understanding of what’s normal for your system. Browse xglance’s online help, the PA manuals, and work from the examples I’ve provided. Have fun! Soon

you'll count yourself among the elite performance "artistes." Together, Glance and the PA provide a well-rounded solution for system performance analysis.

Here are some web references that may be helpful to you:

HP-Software product manuals
HP-Software Support
HP-UX Knowledge On Demand
Event and Performance Management
Operations Center solutions

You are welcome to send the author feedback on this paper.

Revision: 10JUN08 dougg