HP-UX Stress Testing: Improving Customer Experience

Pam Holt, Senior Technical Consultant

February, 2004

Presentation Overview

- HP Quality Organization Overview & Stress Test Function
- Problem Overview: Doing the appropriate amount of stress testing for a time to market driven organization.
  - Development & Program Management vs. Stress Testing
  - How to measure stress level?
  - How to compare to customer environments?
  - How to talk to management about stress testing?
- Workload Data Overview & Analysis
- Streamlining Workload Data Presentation
  - Developer view
  - Manager view
  - Comparing 2 systems or runs
- Conclusions & Next Steps

HP-UX Quality Overview: Mission

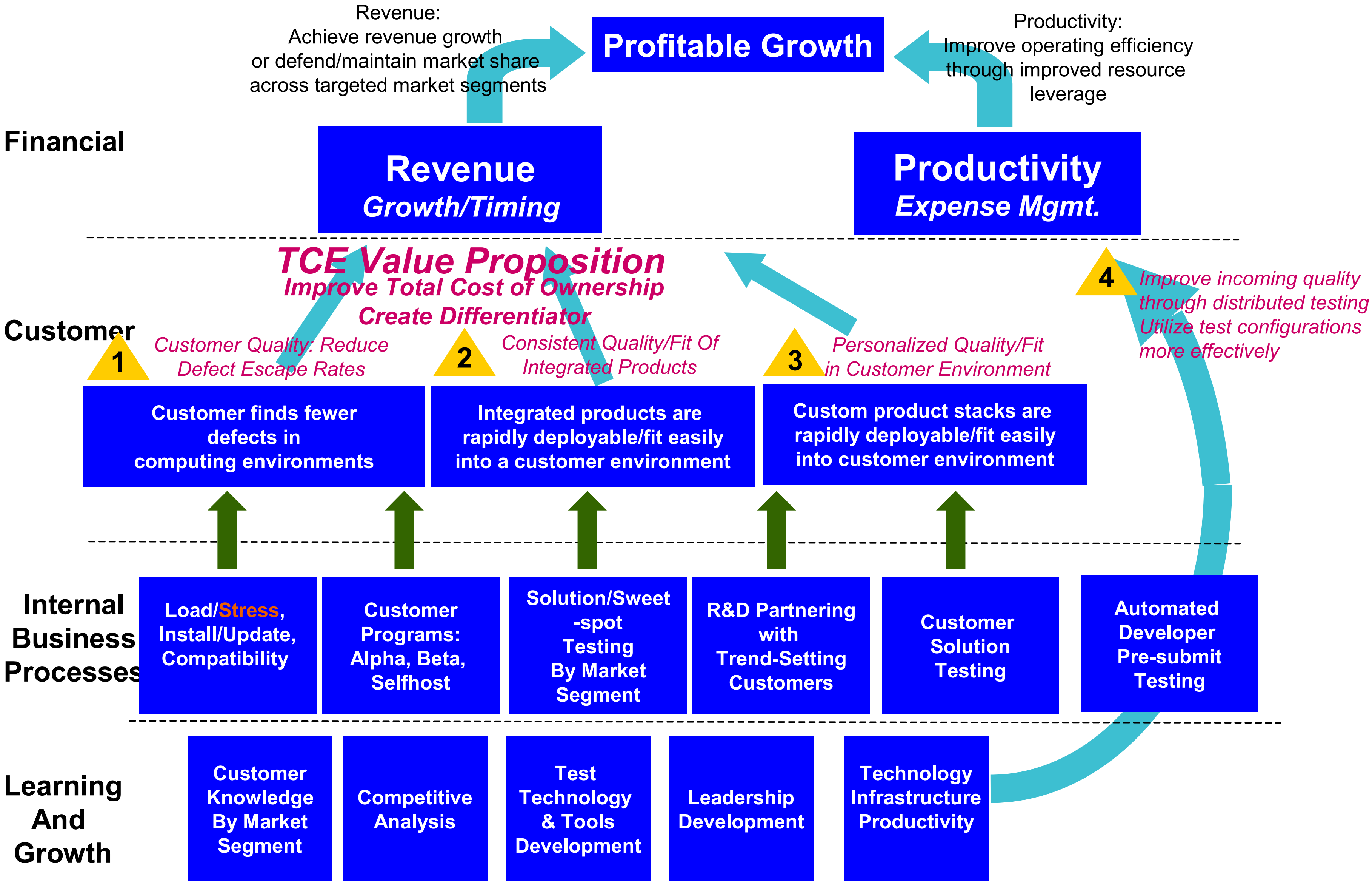
"Providing the Highest Quality Operating Environments for the Always On Infrastructure"

- Continuously improving end-to-end quality
- Maintaining compatibility
- Ensuring rapid time-to-solutions for HP-UX operating environments from a customer's perspective

HP-UX Test Overview: Test Methods

- System level testing on customer-like configurations for multiple test types including stress, install/update, functional, Open Group standards' compliance, compatibility, and usability.
  - Testing takes place on configurations ranging from single CPUs to large memory and maximum CPU prototypes, as well as on multi-tier/multi OS environments with third party products including Oracle, SAP and SAS.
- Work directly with trend setting customers to validate the performance and stability of HP-UX in their dynamic environments.

Test Strategy Map



Revenue:
Achieve revenue growth
or defend/maintain market share
across targeted market segments

Profitable Growth

Productivity:
Improve operating efficiency
through improved resource
leverage

Financial

Revenue
Growth/Timing

Productivity
Expense Mgmt.

TCE Value Proposition
Improve Total Cost of Ownership
Create Differentiator

Customer

1 *Customer Quality: Reduce Defect Escape Rates*

2 *Consistent Quality/Fit Of Integrated Products*

3 *Personalized Quality/Fit in Customer Environment*

4 *Improve incoming quality through distributed testing*
Utilize test configurations more effectively

Customer finds fewer defects in computing environments

Integrated products are rapidly deployable/fit easily into a customer environment

Custom product stacks are rapidly deployable/fit easily into customer environment

Internal Business Processes

Load/Stress, Install/Update, Compatibility

Customer Programs: Alpha, Beta, Selfhost

Solution/Sweet-spot Testing By Market Segment

R&D Partnering with Trend-Setting Customers

Customer Solution Testing

Automated Developer Pre-submit Testing

Learning And Growth

Customer Knowledge By Market Segment

Competitive Analysis

Test Technology & Tools Development

Leadership Development

Technology Infrastructure Productivity

Problem Statement:

- Classic push-pull between Time to Market pressures and Quality
- Defects found in Stress testing are hard to debug, hard to reproduce, hard to root cause
  - Stress testing is a good method for finding a wide range of corner case defects, but it is difficult to differentiate between corner case and customer visible defects.
  - Internal development partners felt stress levels were unrealistically high. This caused them to discount stress defects.
  - Stress test groups and support wanted to ensure minimal defect escapes to customers.
- What is “right” amount of Stress testing to meet customer needs?
- What are “right” defects to fix– i.e. defects customers will see?

Solutions:

- Understand Customer found defects:
  - Analyze defects that escape to customers & feed test recommendations back to appropriate place in test continuum.
  - Plug holes.
- Understand Customer Operational Profile (OP) and Increase Realism in Development & Test:
  - Gather customer operational profile (OP: configuration and workload data).
  - Compare customer OP's to stress test configurations and workloads.
  - Educate testers, development partners and release management.

How to Analyze & Compare OP’s

> “Showing complexity is hard work.”
>
> – Edward Tufte

Tell the truth.

Show the data in its full complexity.

Reveal what is hidden.

How to gather customer workload data?

- What data do we gather?
  - How do we measure workloads/stress internally?
  - How do we compare different workloads?
  - How do customers monitor their workloads?
  - What tools are available to measure workloads in customer production environments?
- Common language was needed to characterize workloads.
  - We needed to compare internal test workload data and customer workload data.
  - We needed to minimize impact to customer production environments in our data gathering.
  - We needed to use tools that would work in pre-release test environments.

Conflicting Requirements!

Internal Tool Requirements	**Customer Tool Requirements**
Tools must work on pre-release and sometimes unstable systems.	Tools must not perturb production environments.
Test was focused on tracking kernel resource utilization and or unit coverage.	Customers measure ability to get a job done, not usage of a particular kernel resource.
Homegrown tools OK—better, in fact!	Fully tested released tools needed!
Reality: Downstream partners produce stable tools too late for use by system test.	**Reality:** Customers won't change their practices just to provide HP with workload data.

Measurements and Tools we investigated:

Top

- Load average: Our “traditional” measure of stress. The load average is maintained by the kernel. It is defined as an average of the number of processes running and ready to run, as sampled over the previous one-minute interval of system operation.
- Did not meet the need of large systems
- Did not represent the depth of resource utilization on a system

OpenView Tools: Glance+ Pack, MeasureWare, Vantage Point Performance Agent, PerfView

SAR

Accounting

Home Grown tools

Glance+ / MeasureWare / Vantage Point Performance Agent

Common “Measurement Dictionary” spanned the range of perspectives from resource focus to task focus to solution focus.

Glance+ available early in release.   Scripts can automate collecting statistics over length of test runs.

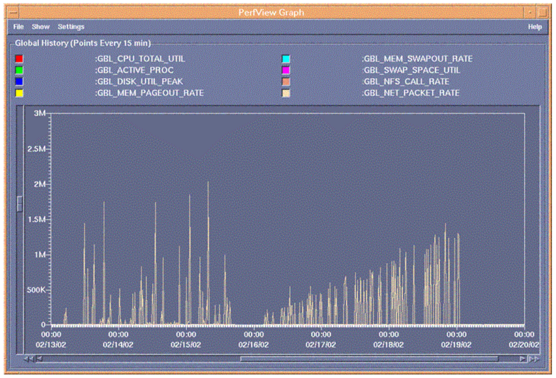
MeasureWare available later but was used by the field to monitor system performance. Most customers use it.

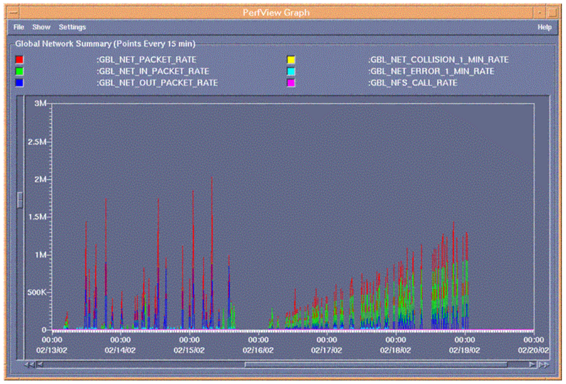
- We could compare internally and externally generated data without impacting customer operations.

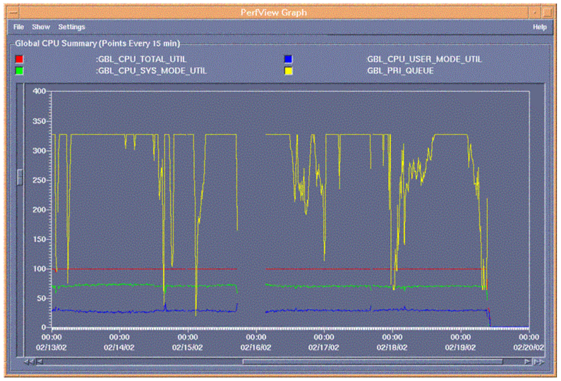
MeasureWare Data Walkthrough:

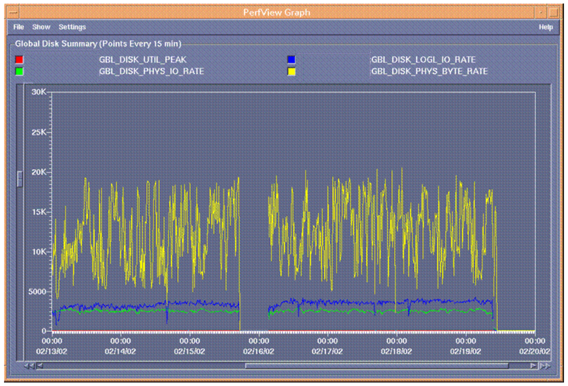
- Eye Test—overview of 6 MW graphs
- Backup slides contain review & commentary on contents of each graph: Global History, Networking, CPU, Disk, Memory, Queue
- Review spider chart summary of the data
- Map spider data to MeasureWare Graphs
- Review of spider chart comparison graphs

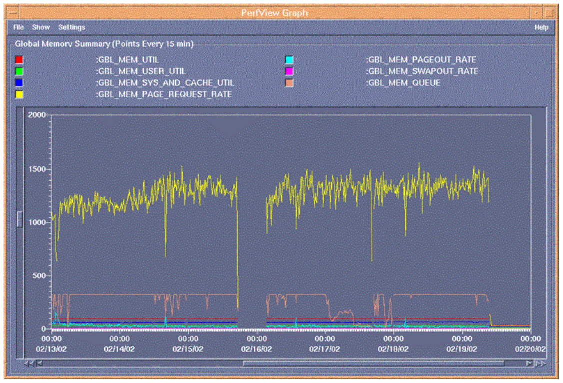
MeasureWare Graph Overview: Eye Test

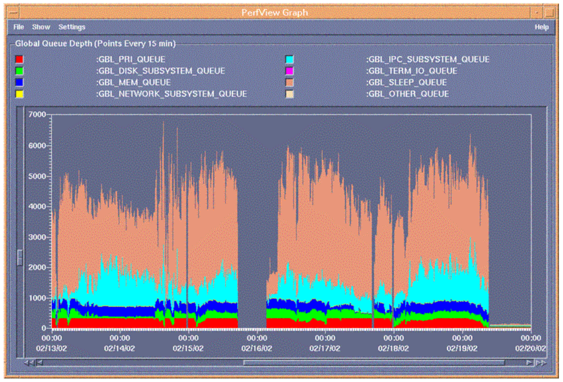








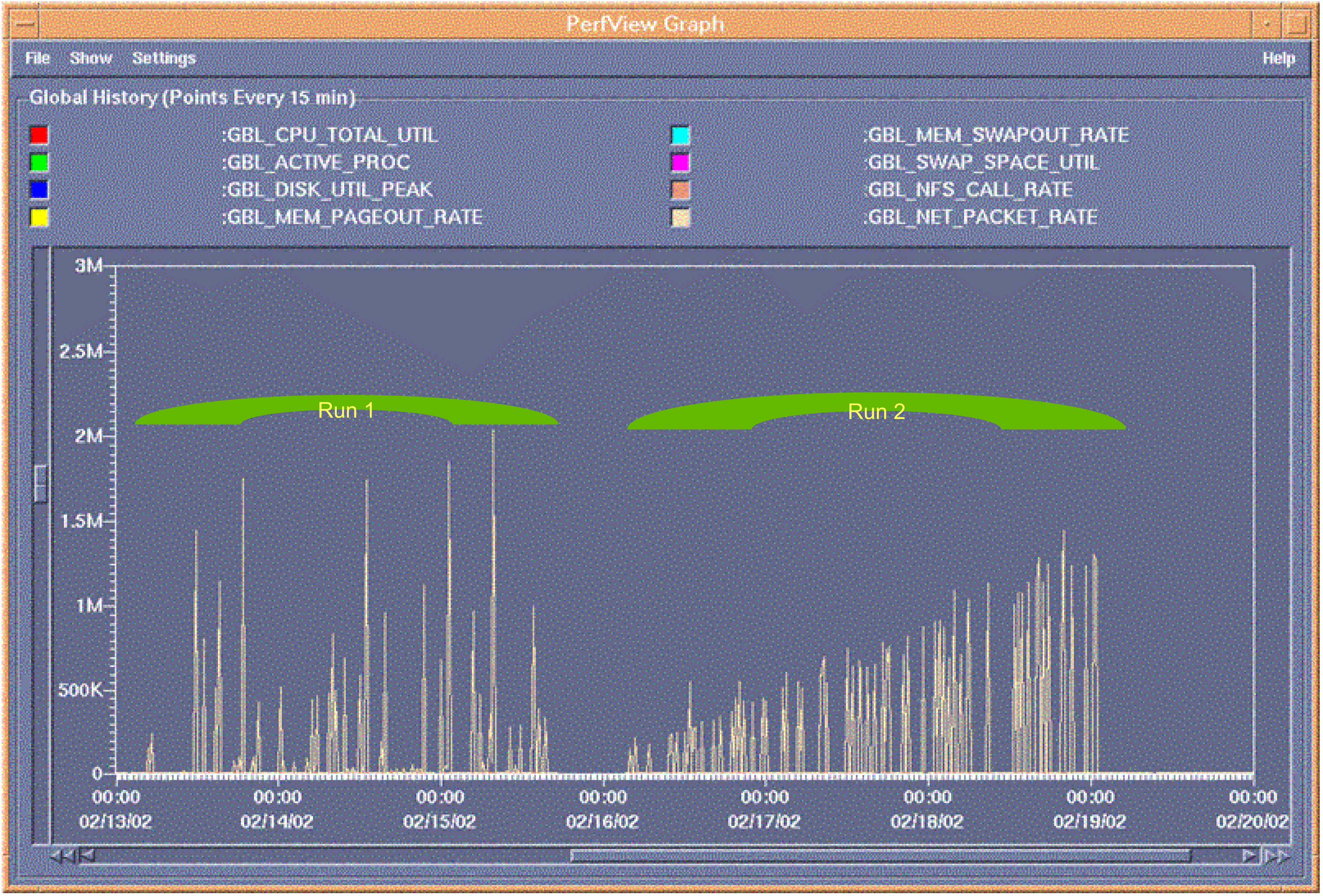




Too much data to analyze!

- 400+ MeasureWare Data points
- Also can get a view of application resource utilization (if applications are configured with MW)
- Different perspectives on the vast range of workload data.
  - Developers want to drill down on their area of expertise.
    - # System/Library calls, code coverage, table utilization.
  - Managers need high level.
  - Customers just want to solve their business problem with minimal cost.
    - Prefer throughput measures.

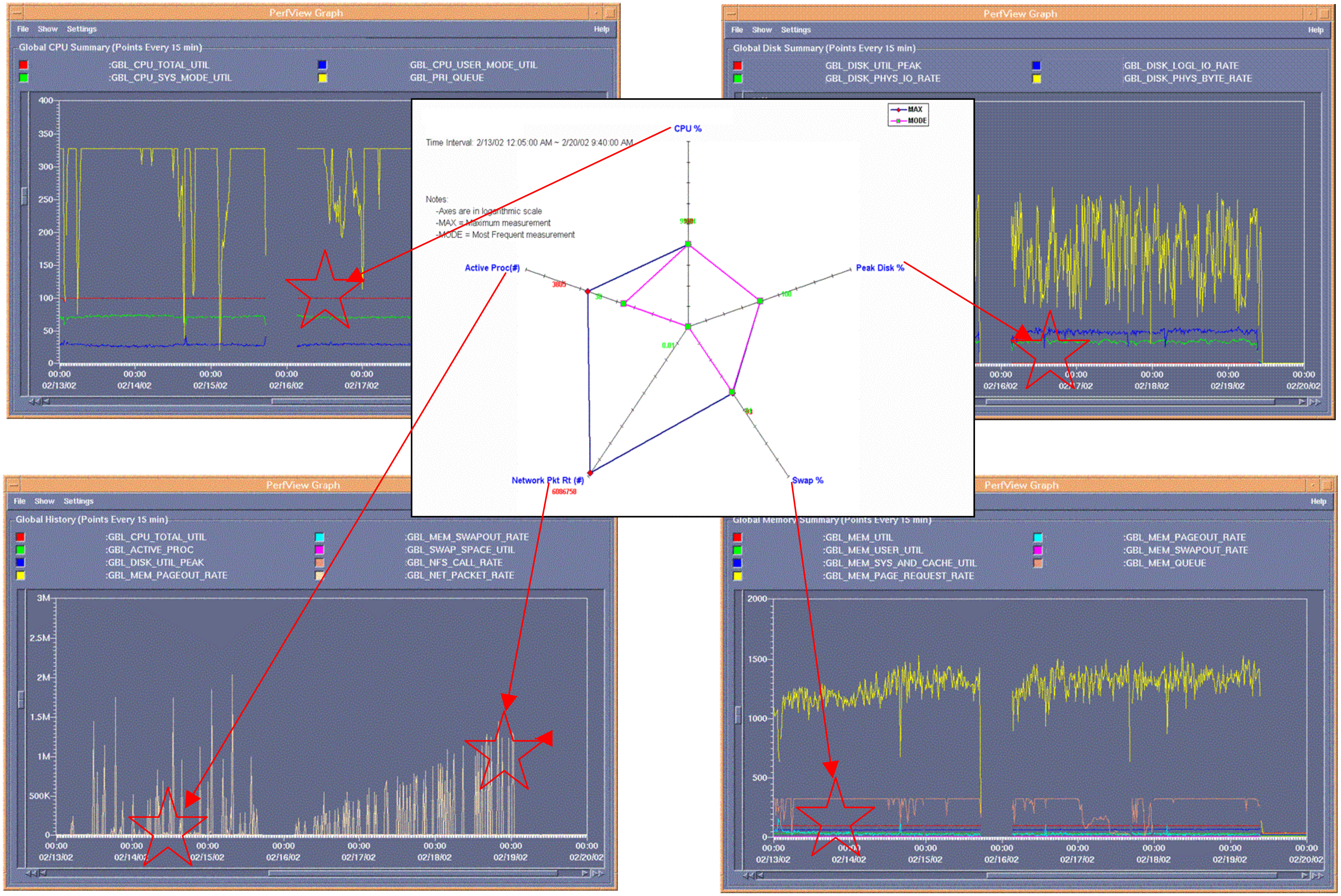
Developer View : Stress Test Global History Graph



MeasureWare Data Walkthrough: Global History Graph

- One week of test data is shown here (use 15 minute averaging for 7 days to allow us to compare graphs)
- System Under Test (SUT) ran 2 24 hour runs, with a short cleanup period in between. Run 1 ended with a panic. Run 2 ran to completion. The light brown line is the network traffic rate for all network cards on the SUT.
- All other data is dwarfed by the high level of network traffic
- The second test run increased the network traffic over the test run
- ADDITIONAL DETAIL ON MEASUREMENTS is found on backup slides.

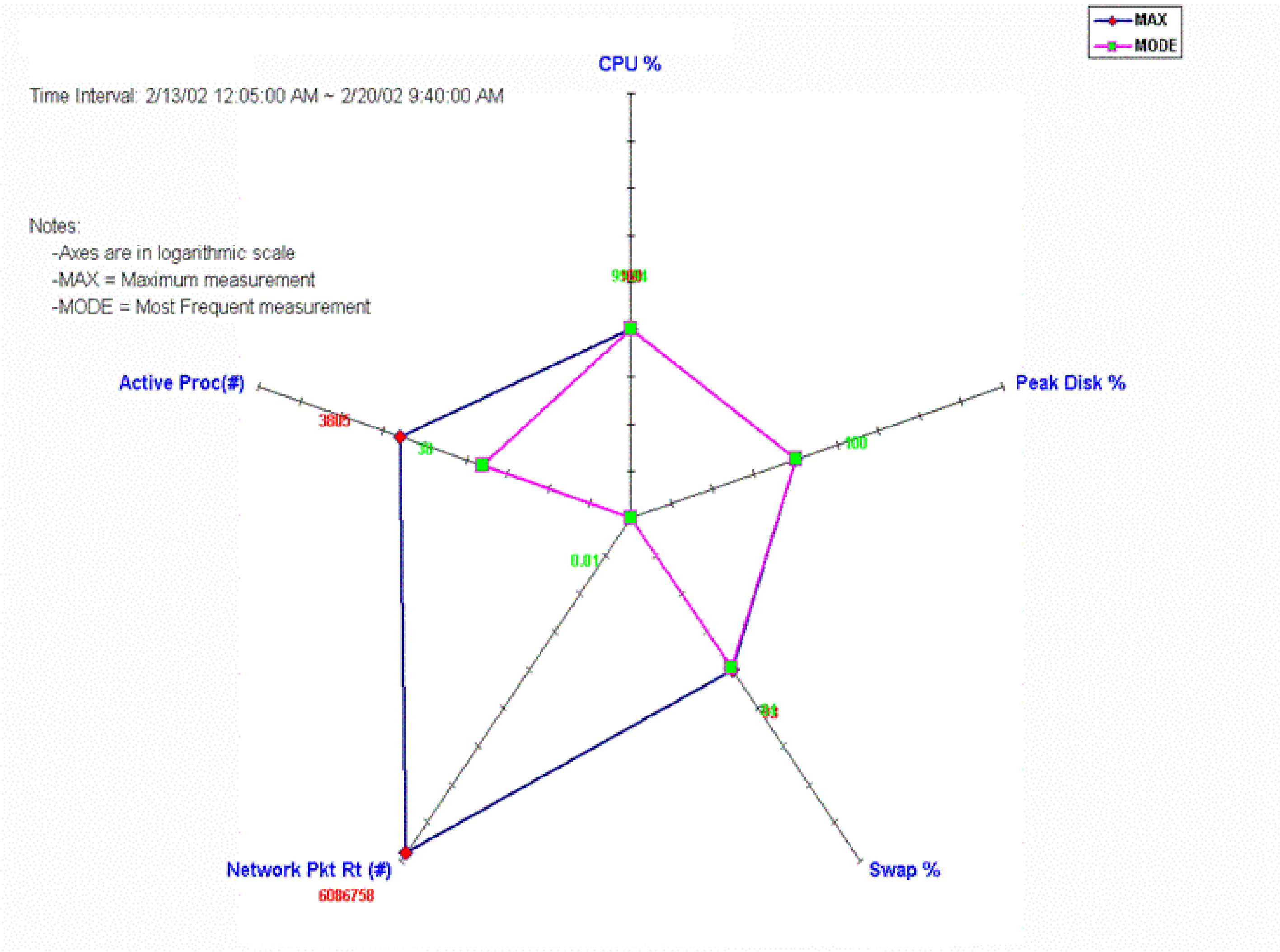
Manager View:
Stress Test Spider Graph: Eye Test



Manager View of Workload Data

- Spider Chart depicts 5 key areas of system Workload data
  - Total CPU %
  - Peak Disk %
  - Swap%
  - Network
  - Active Processes
- Displayed Maximum and Modal values for each
- Load Average was a common measure used by test engineers. We did not feel that was useful enough to include.

Stress Test Spider Chart Overview
Audience: Management



Time Interval: 2/13/02 12:05:00 AM ~ 2/20/02 9:40:00 AM

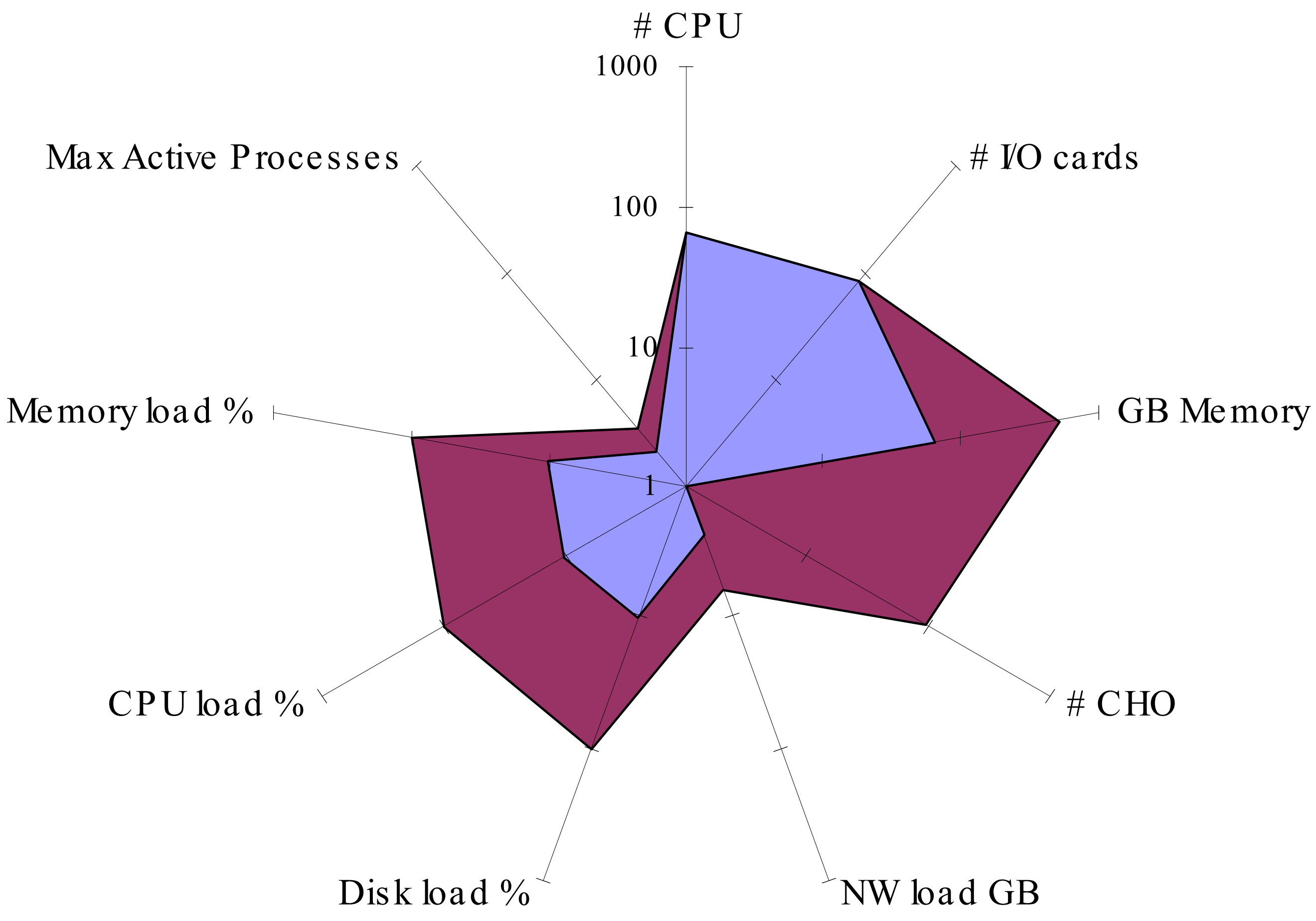
Notes:

- -Axes are in logarithmic scale
- -MAX = Maximum measurement
- -MODE = Most Frequent measurement

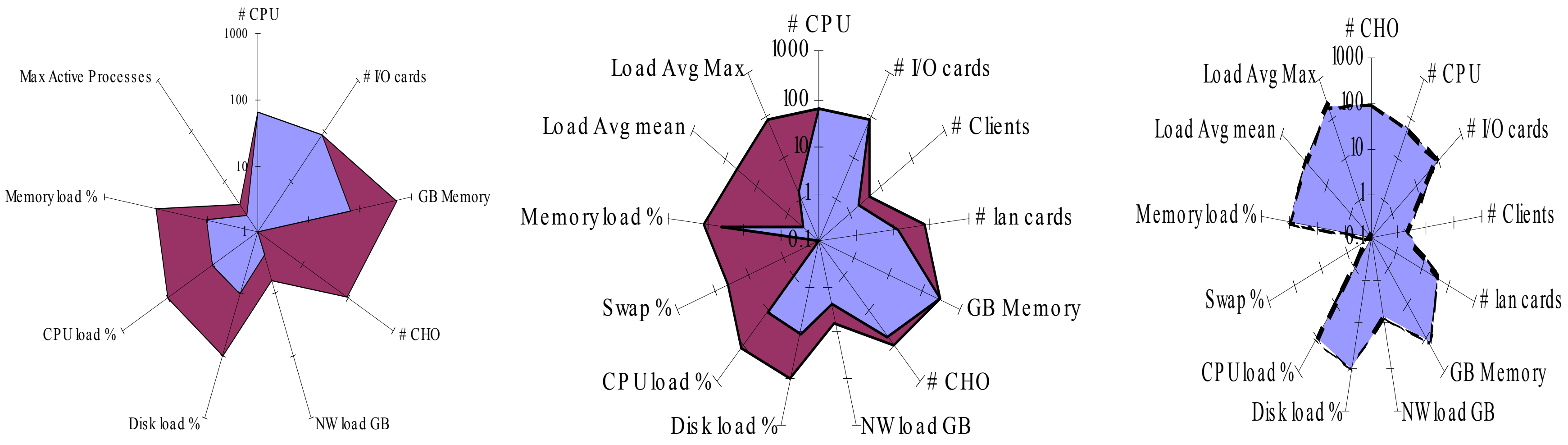
Communicating Pre-Release Test Progress

- Problem: Pre-release test had problems attaining configuration size and load levels
- Solution: Build on Tufte's concept of small multiples to watch a system progress to release quality
  - We expanded the spider graphs to depict configuration targets vs. attained as well as a larger variety of workload targets vs. attained
  - We derived our targets from the prior release test efforts
  - We compared our targets with available customer data to understand realism of test loads

Stress Test Target vs. Attained: 1



Stress Test Target vs. Attained: over time



- Tufte’s small multiples show progress toward goals from beginning of large configuration test to final, release-quality run.
- These graphs mixed configuration goals (#CPU, #IO Cards, GB Memory, # Clients), and runtime goals to highlight early problems attaining release configuration and later load goals.
- Presentation would have been improved if we had kept graphs the same over time.

Conclusions:

- Spider charts are a useful tool to communicate workload data
  - Data reduction from 400 mw data points →30 Global graph data points →5-10 data points on Spider charts (ala Tufte's Small Multiples) helped if targeted to the correct audience & supported by careful analysis.
  - Did we test as well as last release? Extremely valuable in comparing similar test runs to see differences in max and modal values.
  - Not as good at comparing dissimilar test runs. Need Line graphs as backup.

Challenges

- Aren’t you done testing yet?
  - Testers & some managers feared that spider charts would be used to predict ship date.
- Engineering team found problems that made their numbers look bad:
  - Unstable pre-release versions of data gathering tools.
  - Defects in key subsystems prevented engineers from fully scaling the test load.
- Needed to take care in setting target values especially on new hardware.
  - High end systems could never reach targets for swap space utilization based on midrange systems.
  - A kernel limitation on high end systems together with increased throughput on those machines prevented reaching Active Process targets.
- A “One Size Fits All” spider graph does not reflect test objectives.

Next Steps

- Ideal situation would be to tailor the spider chart to match the test objective.
  - In a complex release, multiple test rings with multiple test objectives and multiple star charts would be needed.
  - Now that we better understand the usage statistics, we can move back to one number that would summarize goals vs. achieved.
- Engineers should use spider charts during test run to monitor effectiveness.
- How would we compare load on Tru64 system or a Linux system?
- Continually educate testers & managers about importance of load monitoring.
- Continually gather and analyze customer OP data & feed back to development & test community.

Additional information

MeasureWare & Glance:

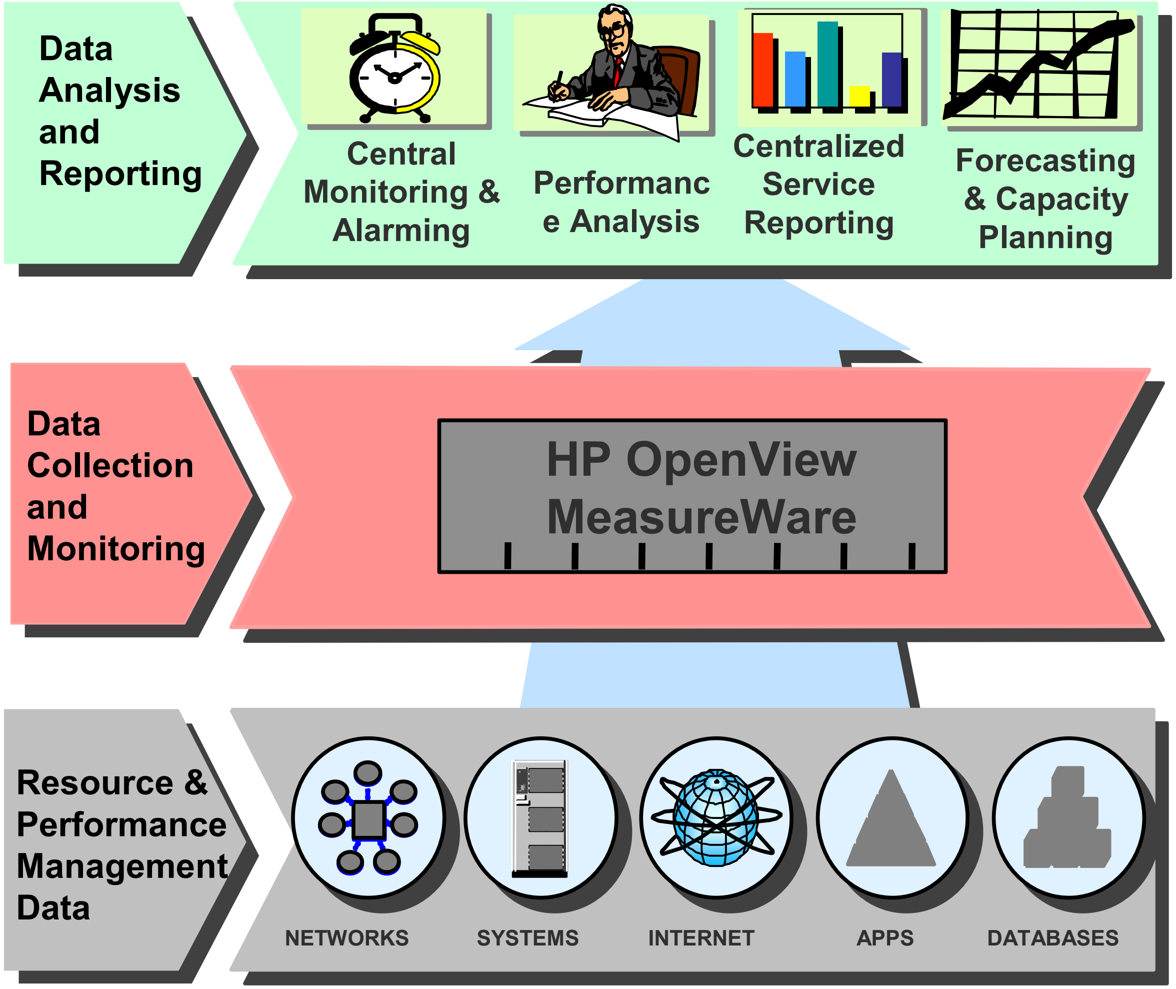
http://www.openview.hp.com/

Edward R. Tufte

Envisioning Information (1990)
Visual Explanations (1997)
The Visual Display of Quantitative Information (2001)
http://www.edwardtufte.com

Background Slides

GlancePlus Pak 2000
- MeasureWare

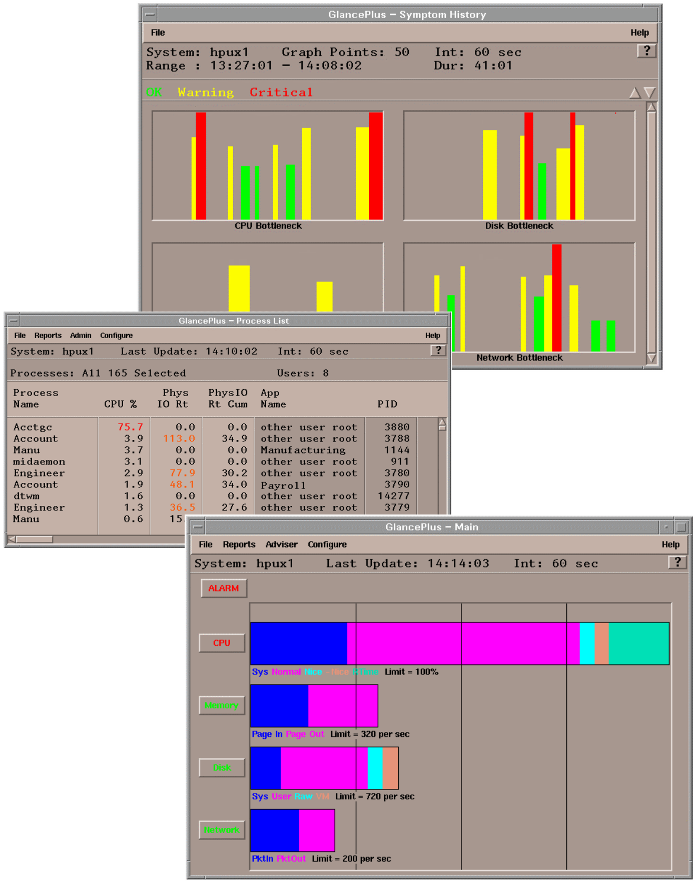

Data Analysis and Reporting
Central Monitoring & Alarming
Performanc e Analysis
Centralized Service Reporting
Forecasting & Capacity Planning
Data Collection and Monitoring
HP OpenView MeasureWare
Resource & Performance Management Data
NETWORKS
SYSTEMS
INTERNET
APPS
DATABASES

- Low overhead collection technology
- Powerful alarming
- Open interfaces on both sides
- OpenView Building Block Architecture
- ARM compliance

GlancePlus Pak 2000
What is GlancePlus?

The premier system diagnostic tool on Open Systems!

- With more than 100,000 copies sold, it is the most widely used Unix diagnostic tool in the market
- It is a fire-tested tool that has grown and flourished from its several years of use
- Its awarding-winning Help and its easy-to-use design - allow your IT personnel to quickly get productive


GlancePlus – Symptom History
File
Help
?
OK
Warning
Critical
CPU Bottleneck
Disk Bottleneck
Network Bottleneck
GlancePlus – Process List
File
Reports
Admin
Configure
Help
?
GlancePlus – Main
File
Reports
Adviser
Configure
Help
?
ALARM
CPU
Memory
Disk
Network
Sys Normal Nice -Nice RTime Limit = 100%
Page In Page Out Limit = 320 per sec
Sys User Raw VM Limit = 720 per sec
PktIn PktOut Limit = 200 per sec

```
System: hpux1     Graph Points: 50    Int: 60 sec
Range : 13:27:01 – 14:08:02           Dur: 41:01
```

```
System: hpux1     Last Update: 14:10:02    Int: 60 sec

Processes: All 165 Selected                Users: 8

Process               Phys    PhysIO  App
Name         CPU %    IO Rt   Rt Cum  Name                  PID

Acctgc        75.7      0.0     0.0   other user root      3880
Account        3.9    113.0    34.9   other user root      3788
Manu           3.7      0.0     0.0   Manufacturing        1144
midaemon       3.1      0.0     0.0   other user root       911
Engineer       2.9     77.9    30.2   other user root      3780
Account        1.9     48.1    34.0   Payroll              3790
dtwm           1.6      0.0     0.0   other user root     14277
Engineer       1.3     36.5    27.6   other user root      3779
Manu           0.6     15
```

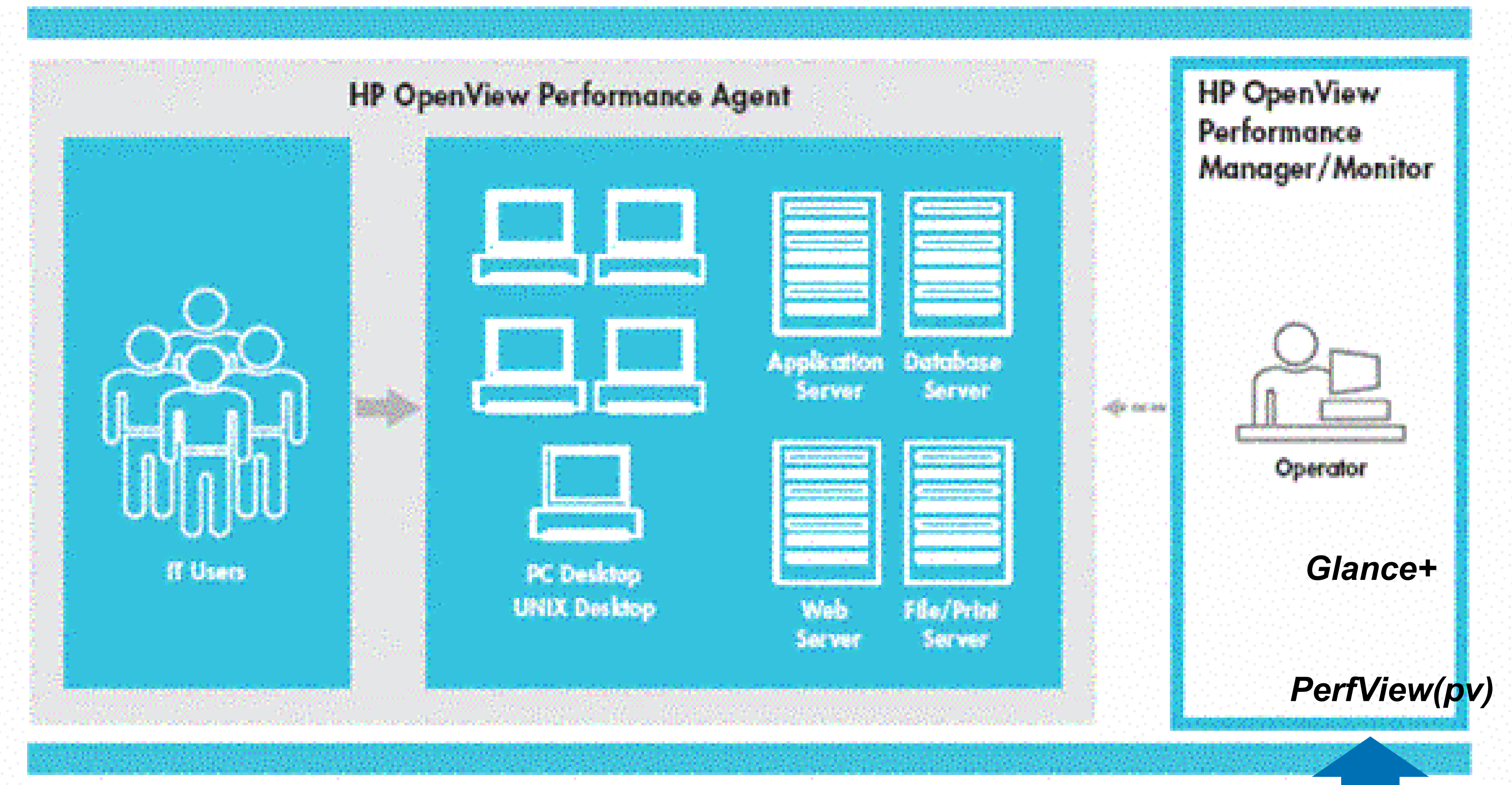
```
System: hpux1     Last Update: 14:14:03    Int: 60 sec
```

March 2004

GlancePlus

- GlancePlus provides in-depth, real-time troubleshooting and monitoring of individual systems, providing the best possible performance from a computer system. GlancePlus works standalone or in conjunction with PerfView/MeasureWare.
- Benefits:
  - Available early in release.
  - Allows drill down on specific kernel components.
  - 400+ data points
- Drawbacks:
  - No Graphs, but they can be created with additional work

OpenView/MeasureWare Overview



Glance+

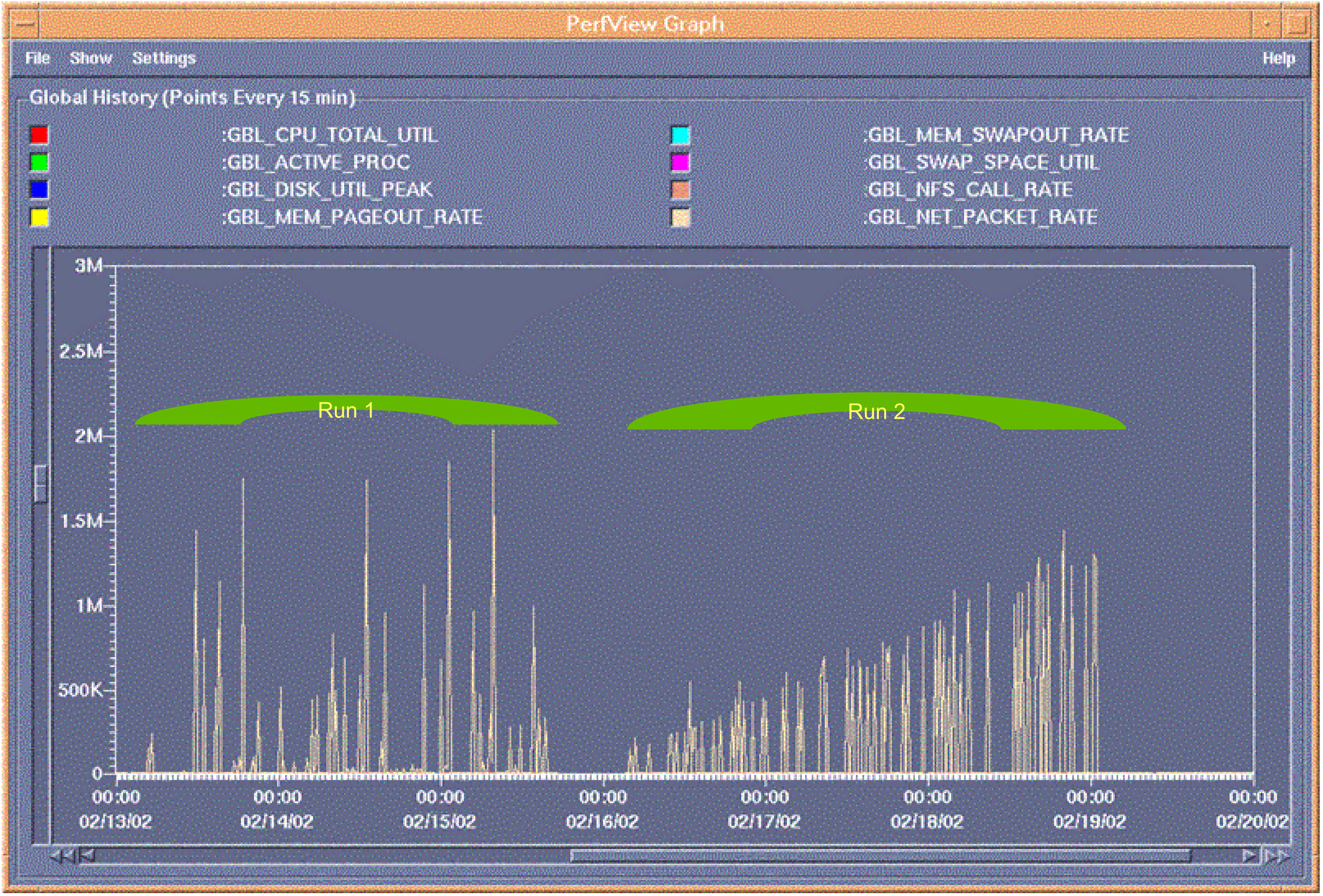
PerfView(pv)

Measureware Dictionary

MeasureWare/PerfView example

- PerfView provides a single interface and a common method for centrally monitoring, analyzing, comparing, and forecasting IT resource utilization measurement data. PerfView is available for the HP-UX and Windows NT platforms.
- Benefits:
  - **Graphs with different views of workload.**
  - **Graphs customizable.**
  - **Field Support team asks customers to run MW.**
  - **Allows drill down on specific kernel components.**
  - **Allows solution view if customer wants to configure it.**
- **Drawbacks:**
  - **Available late in release.**
  - **Some problems running tool under high stress.**

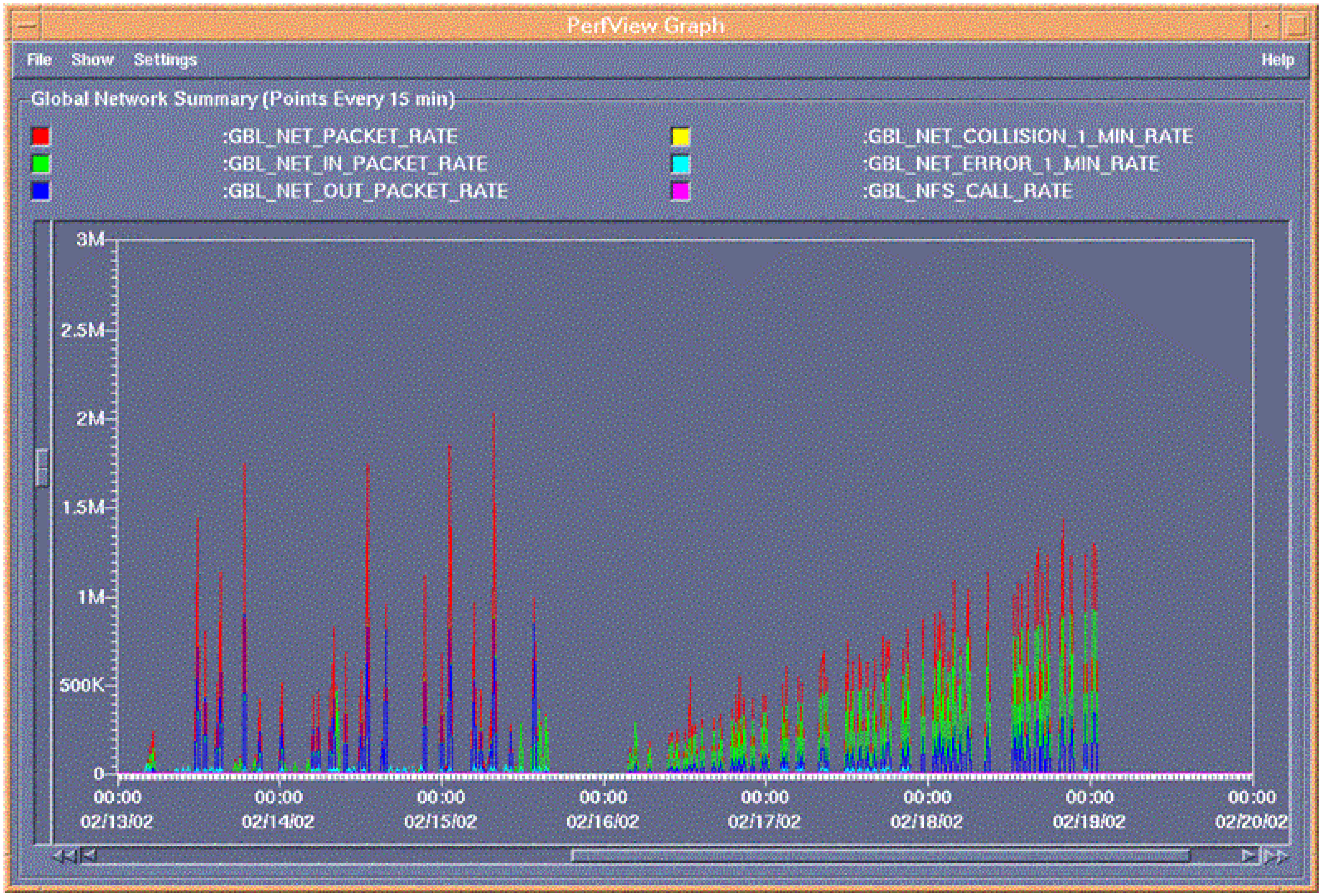
Stress Test Global History Graph



MeasureWare Data Walkthrough: Global History Graph

- One week of test data is shown here (use 15 minute averaging for 7 days to allow us to compare graphs)
- System Under Test (SUT) ran 2 24 hour runs, with a short cleanup period in between. Run 1 ended with a panic. Run 2 ran to completion. The light brown line is the network traffic rate for all network cards on the SUT.
- All other data is dwarfed by the high level of network traffic
- The second test run increased the network traffic over the test run

Stress Test Global Network Graph



MeasureWare Data Walkthrough: Global Network Graph

- Shows more detail of the network subsystem: Input, output, total packets

MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NET_PACKET_RATE**
- GBL_NET_PACKET_RATE is the number of successful packets per second (both inbound and outbound) for all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.
- On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_INTERVAL can be greater than the sampling interval.

MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NET_IN_PACKET_RATE**
- The number of successful packets per second received through all network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.
- On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_INTERVAL can be greater than the sampling interval.

- **GBL_NET_OUT_PACKET_RATE**
- The number of successful packets per second sent through the network interfaces during the interval. Successful packets are those that have been processed without errors or collisions.
- On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_INTERVAL can be greater than the sampling interval.

MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NET_COLLISION_1_MIN_RATE**
- The number of collisions per second on all network interfaces during the interval.
- A rising rate of collisions versus outbound packets is an indication that the network is becoming increasingly congested.
- On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_INTERVAL can be greater than the sampling interval.

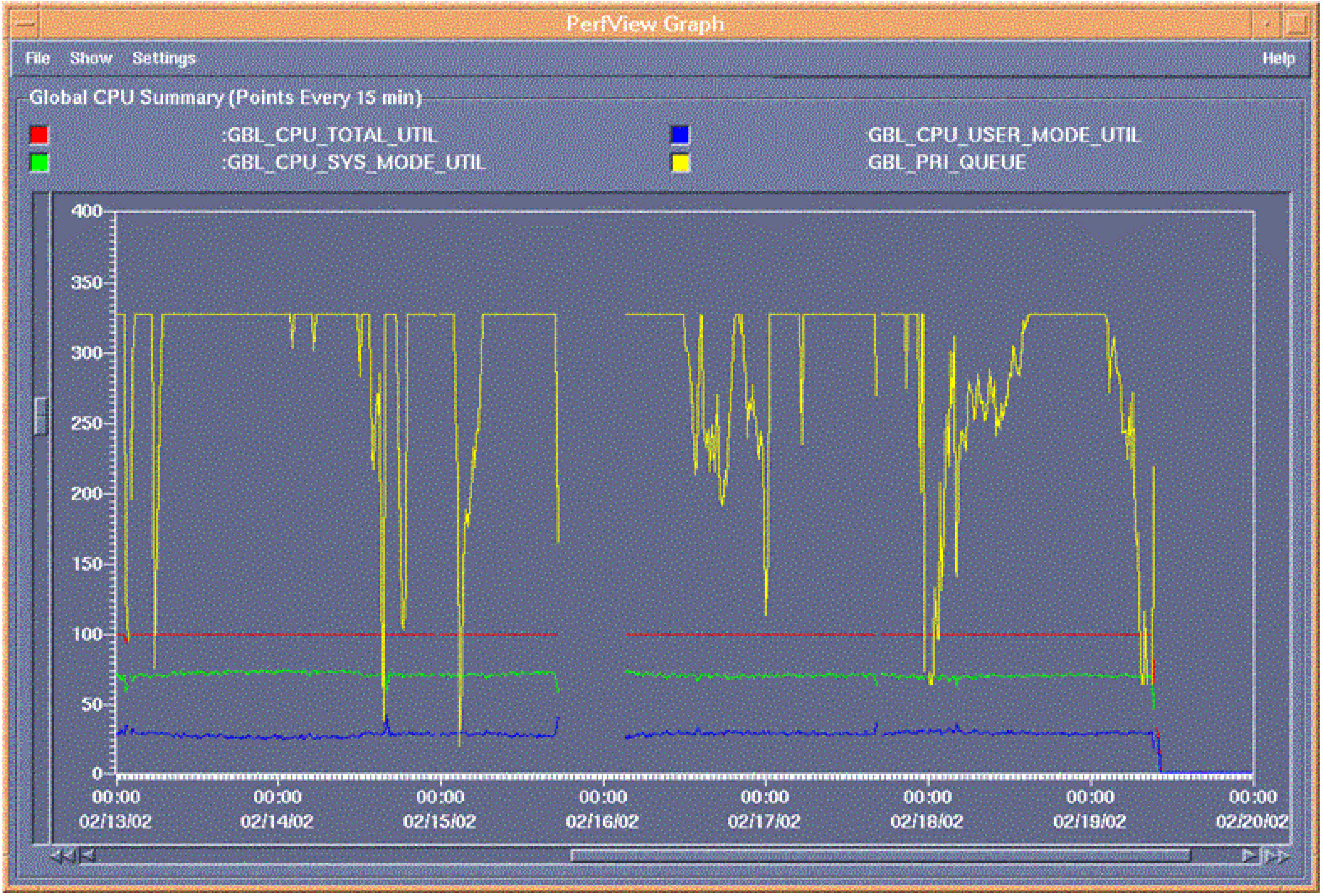
MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NET_ERROR_1_MIN_RATE**
- The number of errors per minute on all network interfaces during the interval. This rate should normally be zero or very small. A large error rate can indicate a hardware or software problem.
- On HP-UX 11.0 and beyond for Glance and GPM, this metric is updated at the BYNETIF_INTERVAL time. On systems with large numbers of IP addresses, the BYNETIF_INTERVAL can be greater than the sampling interval.

MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NFS_CALL_RATE**
- The number of NFS calls per second the system made as either a NFS client or NFS server during the interval.
- Each computer can operate as both a NFS server, and as an NFS client.
- This metric includes both successful and unsuccessful calls. Unsuccessful calls are those that cannot be completed due to resource limitations or LAN packet errors.
- NFS calls include create, remove, rename, link, symlink, mkdir, rmdir, statfs, getattr, setattr, lookup, read, readdir, readlink, write, writecache, null and root operations.

Stress Test Global CPU Graph



MeasureWare Data Walkthrough: Global CPU Graph

- **GBL_CPU_TOTAL_UTIL**
- This metric represents the percentage of time the CPU was not idle during the interval.
- CPU total utilization is the sum of system mode utilization plus user mode utilization.
- GBL_CPU_TOTAL_UTIL = GBL_CPU_USER_MODE_UTIL + GBL_CPU_SYSTEM_MODE_UTIL
- GBL_CPU_TOTAL_UTIL + GBL_CPU_IDLE_UTIL= 100%
- On a system with multiple CPUs, this metric is normalized by dividing the CPU utilization over all processors by the number of processors online. This represents the usage of the total processing capacity available.

MeasureWare Data Walkthrough: Global CPU Graph

- **GBL_CPU_SYS_MODE_UTIL**
- Percentage of time the CPU was in system mode during the interval.
- This metric is a subset of the GBL_CPU_TOTAL_UTIL percentage.
- A UNIX process operates in either system mode (also called kernel mode) or user mode. When a process requests services from the operating system with a system call, it switches into the machine's privileged protection mode and runs in system mode.
- This is NOT a measure of the amount of time used by system daemon processes, since most system daemons spend part of their time in user mode and part in system calls, like any other process.
- On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.

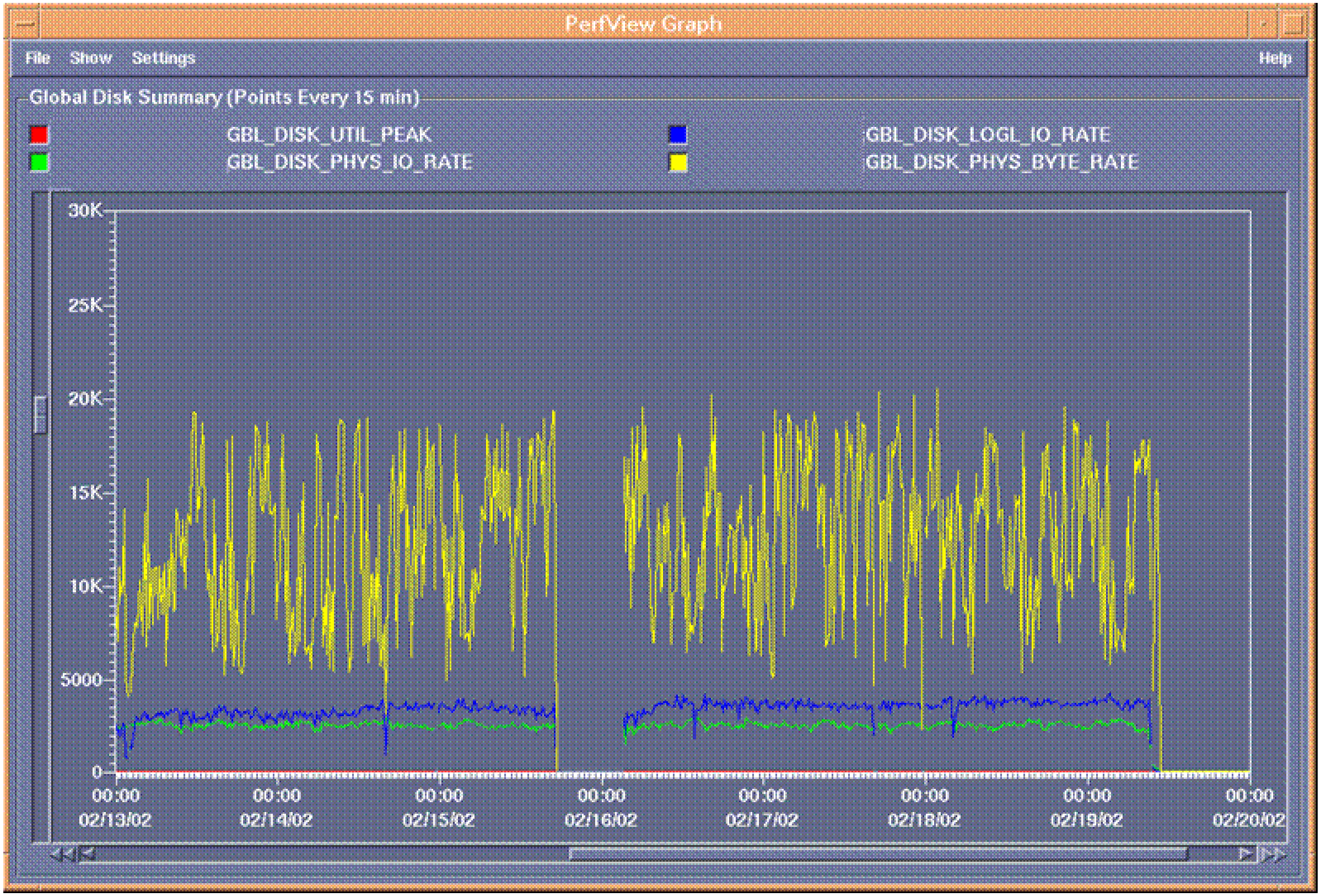
MeasureWare Data Walkthrough: Global CPU Graph

- **GBL_CPU_USER_MODE_UTIL**
- The percentage of time the CPU was in user mode during the interval.
- This metric is a subset of the GBL_CPU_TOTAL_UTIL percentage.
- On a system with multiple CPUs, this metric is normalized. That is, the CPU used over all processors is divided by the number of processors online. This represents the usage of the total processing capacity available.
- High user mode CPU percentages are normal for computation-intensive applications. Low values of user CPU utilization compared to relatively high values for GBL_CPU_SYS_MODE_UTIL can indicate an application or hardware problem.
- User CPU is the time spent in user mode at a normal priority, at real-time priority, and at a nice priority.

MeasureWare Data Walkthrough: Global CPU Graph

- **GBL_PRI_QUEUE**
- The average number of processes or threads blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.
- To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_PRI_QUEUE is greater than three, there is a high probability of a CPU bottleneck.
- This is calculated as the accumulated time that all processes or threads spent blocked on PRI divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

Stress Test Global Disk Graph



MeasureWare Data Walkthrough: Global Disk Graph

- **GBL_DISK_UTIL_PEAK**
- Peak Disk % represents the time, in seconds, during the interval that the ***busiest*** disk was performing IO transfers. This is for the busiest disk only, not all disk devices. This counter is based on an end-to-end measurement for each IO transfer updated at queue entry and exit points. Only local disks are counted in this measurement. NFS devices are excluded. A peak disk utilization of more than 50% often indicates a disk IO subsystem bottleneck situation, which may not be in the physical disk drive itself, but elsewhere in the IO path.

MeasureWare Data Walkthrough: Global Disk Graph

- **GBL_DISK_PHYS_IO_RATE**
- The number of physical IOs per second during the interval.
- Only local disks are counted in this measurement. NFS devices are excluded.
- This includes all types of physical IOs to disk, including virtual memory IO and raw IO.
- This is calculated as
- GBL_DISK_PHYS_IO_RATE =
-     GBL_DISK_FS_IO_RATE +
-     GBL_DISK_VM_IO_RATE +
-     GBL_DISK_SYSTEM_IO_RATE +
-     GBL_DISK_RAW_IO_RATE

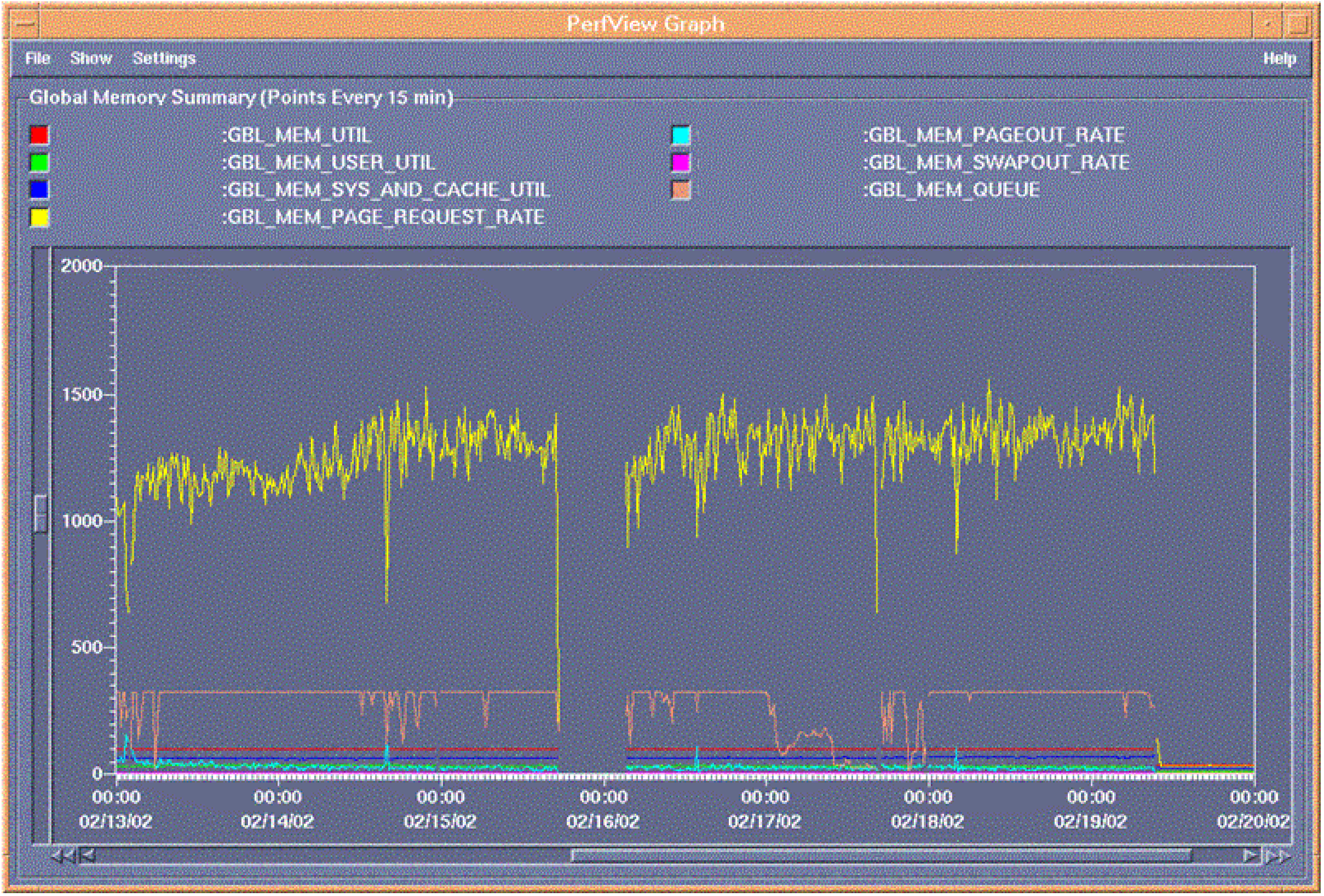
MeasureWare Data Walkthrough: Global Disk Graph

- **GBL_DISK_LOGL_IO_RATE**
- The number of logical IOs per second during the interval.
- Only local disks are counted in this measurement. NFS devices are excluded.
- Logical disk IOs are measured by counting the read and write system calls that are directed to disk devices. Also counted are read and write system calls made indirectly through other system calls, including readv, recvfrom, recv, recvmsg, ipcrecvcn, recfrom, writev, send, sento, sendmsg, and ipcsend.

MeasureWare Data Walkthrough: Global Disk Graph

- **GBL_DISK_PHYS_BYTE_RATE**
- The average number of KB per second at which data was transferred to and from disks during the interval. The bytes for all types physical IO are counted.
- Only local disks are counted in this measurement. NFS devices are excluded.
- This is a measure of the physical data transfer rate. It is not directly related to the number of IOs, since IO requests can be of differing lengths.
- This is an indicator of how much data is being transferred to and from disk devices. Large spikes in this metric can indicate a disk bottleneck.
- This includes file system, virtual memory, and raw IO.

Stress Test Global Memory Graph



MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_UTIL**
- The percentage of physical memory in use during the interval. This includes system memory (occupied by the kernel), buffer cache and user memory.
- This calculation is done using the byte values for physicalmemory and used memory, and is therefore more accurate than comparing the reported kilobyte values for physical memory andused memory.

MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_USER_UTIL**
- The percent of physical memory allocated to user code and data at the end of the interval. This metric shows the percent of memory owned by user memory regions such as user code, heap, stack and other data areas including shared memory. This does not include memory for buffer cache.
- Large fluctuations in this metric can be caused by programs which allocate large amounts of memory and then either release the memory or terminate. A slow continual increase in this metric may indicate a program with a memory leak.

MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_SYS_AND_CACHE_UTIL**
- The percentage of physical memory used by the system (kernel) and the buffer cache at the end of the interval.

MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_PAGE_REQUEST_RATE**
- The number of page requests to or from the disk per second during the interval. This is the same as the sum of the "page ins" and "page outs“ values from the "vmstat -s" command. Remember that "vmstat -s“ reports cumulative counts.
- Higher than normal rates can indicate either a memory or a disk bottleneck. Compare GBL_DISK_UTIL_PEAK and GBL_MEM_UTIL to determine which resource is more constrained. High rates may also indicate memory thrashing caused by a particular application or set of applications. Look for processes with high major fault rates to identify the culprits.

MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_PAGEOUT_RATE**
- The total number of page outs to the disk per second during the interval. This includes pages paged out to paging space and to the file system.
- This is the same as the "page outs" value from the "vmstat -s" command. Remember that "vmstat -s" reports cumulative counts.
- To determine the rate (that is, GBL_MEM_PAGEOUT_RATE) for the current interval, subtract the previous value from the current value and then divide by the length of the interval.
- Keep in mind that whenever any comparisons are made with other tools, both tools must be interval synchronized with each other in order to be valid.

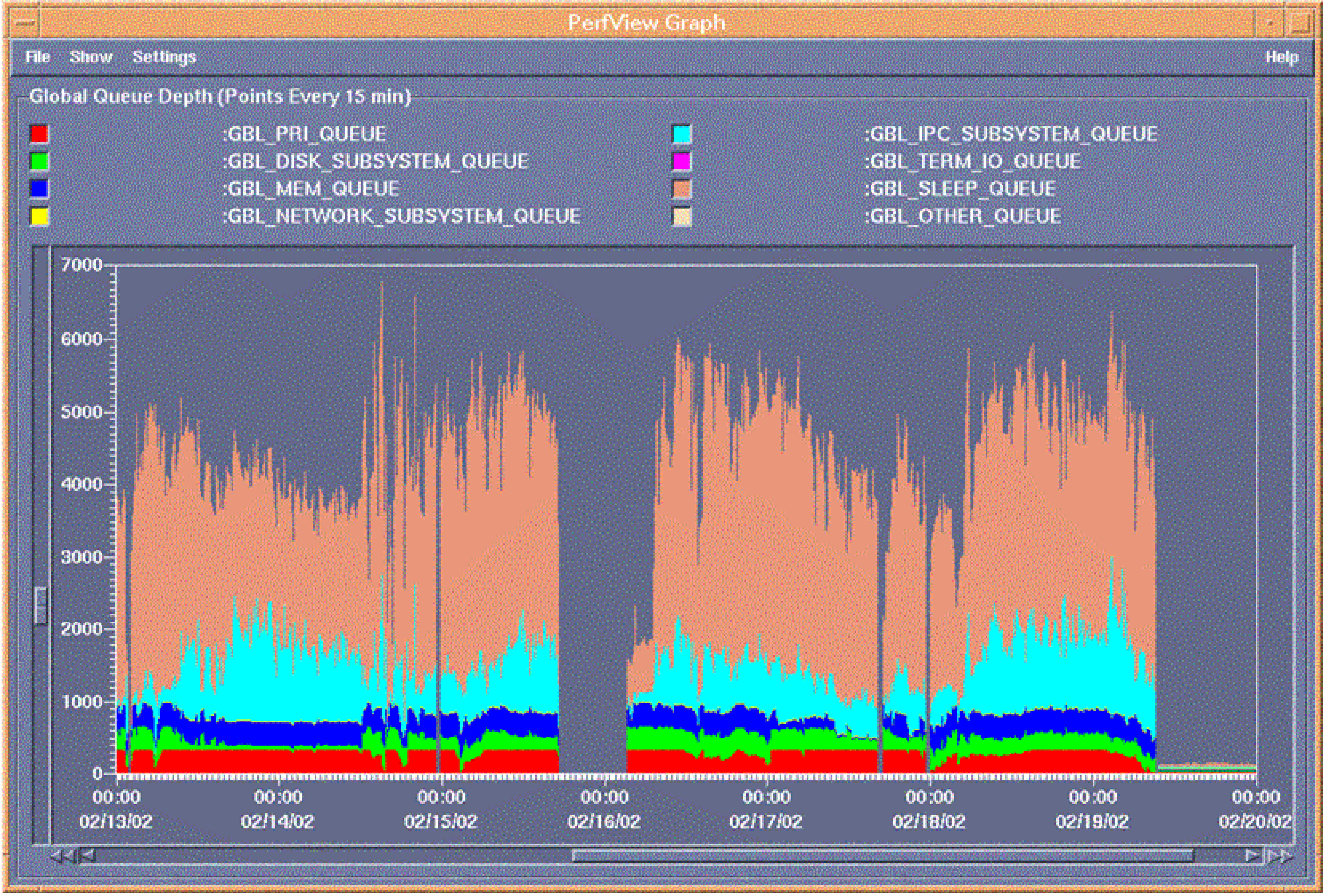
MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_SWAPOUT_RATE**
- The number of deactivations (swap outs prior to HP-UX 10.0) per second during the interval.
- Process swapping has been replaced in HPUX 10.0 by process deactivation. Instead of swapping an entire process to a swap area, processes that place higher demands on memory resources and/or have been inactive for long periods of time are marked as deactivated. Their associated memory regions are deactivated and pages within these regions can be reused or paged out by the memory management vhand process in favor of pages belonging to processes that are not deactivated. Unlike traditional process swapping, deactivated memory pages may or may not be written out to the swap area, because a process could be reactivated before the paging occurs.
- To summarize, a process swap-out in HPUX 10.0 is a process deactivation. A swap-in is a reactivation of a deactivated process. Swap metrics that report swap-out bytes now represent bytes paged out to swap areas from deactivated regions. Because these pages are pushed out over time based on memory demands, these counts are much smaller than 9.0 counts where the entire process was written to the swap area when it was swapped-out. Likewise, swap-in bytes now represent bytes paged in as a result of reactivating a deactivated process and reading in any pages that were actually paged out to the swap area while the process was deactivated.
- This is the same as the "swap outs" value from the "vmstat -s" command. Remember that "vmstat -s" reports cumulative counts.

MeasureWare Data Walkthrough: Global Memory Graph

- **GBL_MEM_QUEUE**
- The average number of processes or threads blocked on memory (waiting for virtual memory disk accesses to complete) during the interval. This typically happens when processes or threads are allocating a large amount of memory. It can also happen when processes or threads access memory that has been paged out to disk (swap) because of overall memory pressure on the system. Note that large programs can block on VM disk access when they are initializing, bringing their text and data pages into memory.
- When this metric rises, it can be an indication of a memory bottleneck, especially if overall system memory utilization (GBL_MEM_UTIL) is near 100% and there is also swapout or page out activity.
- This is calculated as the accumulated time that all processes or threads spent blocked on memory divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

Stress Test Global Queue Graph



MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_PRI_QUEUE**
- The average number of processes or threads blocked on PRI (waiting for their priority to become high enough to get the CPU) during the interval.
- To determine if the CPU is a bottleneck, compare this metric with GBL_CPU_TOTAL_UTIL. If GBL_CPU_TOTAL_UTIL is near 100 percent and GBL_PRI_QUEUE is greater than three, there is a high probability of a CPU bottleneck.
- This is calculated as the accumulated time that all processes or threads spent blocked on PRI divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_DISK_SUBSYSTEM_QUEUE**
- The average number of processes or threads blocked on the disk subsystem (in a "queue" waiting for their file system disk IO to complete) during the interval. This is the sum of processes or threads in the DISK, INODE, CACHE and CDFS wait states. Processes or threads doing raw IO to a disk are not included in this measurement. As this number rises, it is an indication of a disk bottleneck.
- This is calculated as the accumulated time that all processes or threads spent blocked on (DISK + INODE + CACHE + CDFS) divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_MEM_QUEUE**
- The average number of processes or threads blocked on memory (waiting for virtual memory disk accesses to complete) during the interval. This typically happens when processes or threads are allocating a large amount of memory. It can also happen when processes or threads access memory that has been paged out to disk (swap) because of overall memory pressure on the system. Note that large programs can block on VM disk access when they are initializing, bringing their text and data pages into memory. When this metric rises, it can be an indication of a memory bottleneck, especially if overall system memory utilization (GBL_MEM_UTIL) is near 100% and there is also swapout or page out activity.
- This is calculated as the accumulated time that all processes or threads spent blocked on memory divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Network Graph

- **GBL_NETWORK_SUBSYSTEM_QUEUE**
- The average number of processes or threads blocked on the network subsystem (waiting for their network activity to complete) during the interval. This is the sum of processes or threads in the LAN, NFS, and RPC wait states. This does not include processes or threads blocked on SOCKT (that is, sockets) waits, as some processes or threads sit idle in SOCKT waits for long periods.
- This is calculated as the accumulated time that all processes or threads spent blocked on (LAN + NFS + RPC) divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_IPC_SUBSYSTEM_QUEUE**
- The average number of processes or threads blocked on the InterProcess Communication (IPC) subsystems (waiting for their interprocess communication activity to complete) during the interval. This is the sum of processes or threads in the IPC, MSG, SEM, PIPE, SOCKT (that is, sockets) and STRMS (that is, streams IO) wait states.
- This is calculated as the accumulated time that all processes or threads spent blocked on (IPC + MSG + SEM + PIPE + SOCKT + STRMS) divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_TERM_IO_QUEUE**
- The average number of processes or kernel threads blocked on terminal IO (waiting for their terminal IO to complete) during the interval.
- This is calculated as the accumulated time that all processes or kernel threads spent blocked on TERM (that is, terminal IO) divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or kernel thread counts, not actual queues.

MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_SLEEP_QUEUE**
- The average number of processes or threads blocked on SLEEP (waiting to awaken from sleep system calls) during the interval. A process or thread enters the SLEEP state by putting itself to sleep using system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.
- This is calculated as the accumulated time that all processes or threads spent blocked on SLEEP divided by the interval time.
- The Global QUEUE metrics, which are based on block states, represent the average number of process or thread counts, not actual queues.

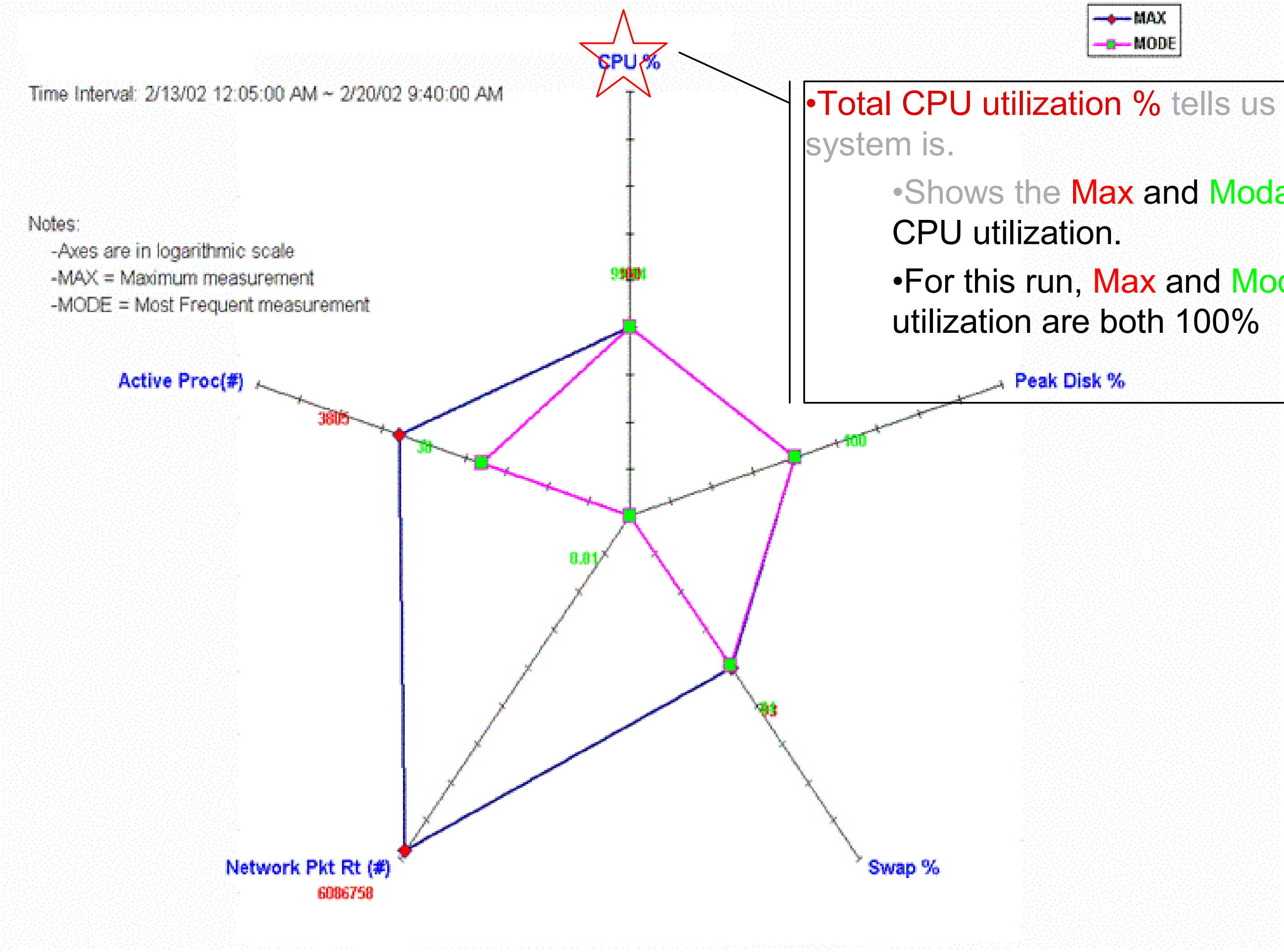
MeasureWare Data Walkthrough: Global Queue Graph

- **GBL_OTHER_QUEUE**
- The average number of processes or kernel threads blocked on other (unknown) activities during the interval.
- This includes processes or kernel threads that were started and subsequently suspended before the midaemon was started and have not been resumed, or the block state is unknown.
- This is calculated as the accumulated time that all processes or kernel threads spent blocked on OTHER divided by the interval time.
- The Global WAIT PCT metrics, which are also based on block states, represent the percentage of all processes or kernel threads that were alive on the system.

Additional Developer Data

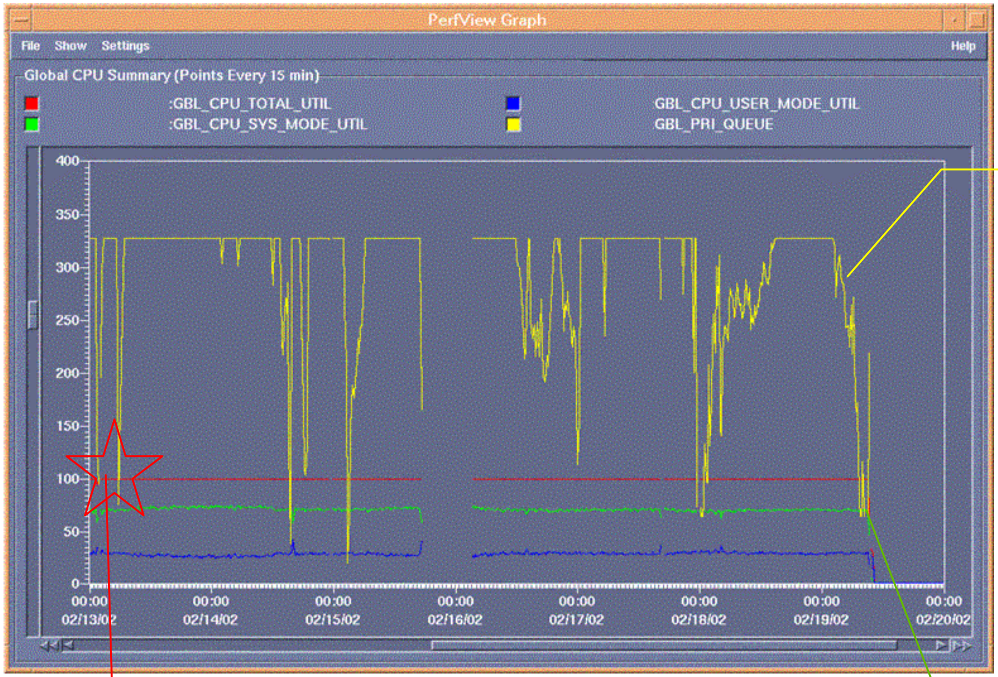
- Can drill down by subsystem to report more focused measurements to help developers get more specific subsystem data.
  - TBL_*
  - PROC_*
  - APP_*
  - BYDSK_*
  - FS_*
  - LV_* (logical volume)
  - BYNETIF_* (by network interface)
  - BYCPU_*
  - SYSCALL_*
  - THREAD_*

Stress Test CPU Utilization



- Total CPU utilization % tells us how busy the system is.
  - Shows the Max and Modal value for Total CPU utilization.
  - For this run, Max and Modal Total CPU utilization are both 100%

Stress Test CPU Utilization



•Priority Queue is seldom greater than 10 on customer systems. This indicates a CPU bottleneck.

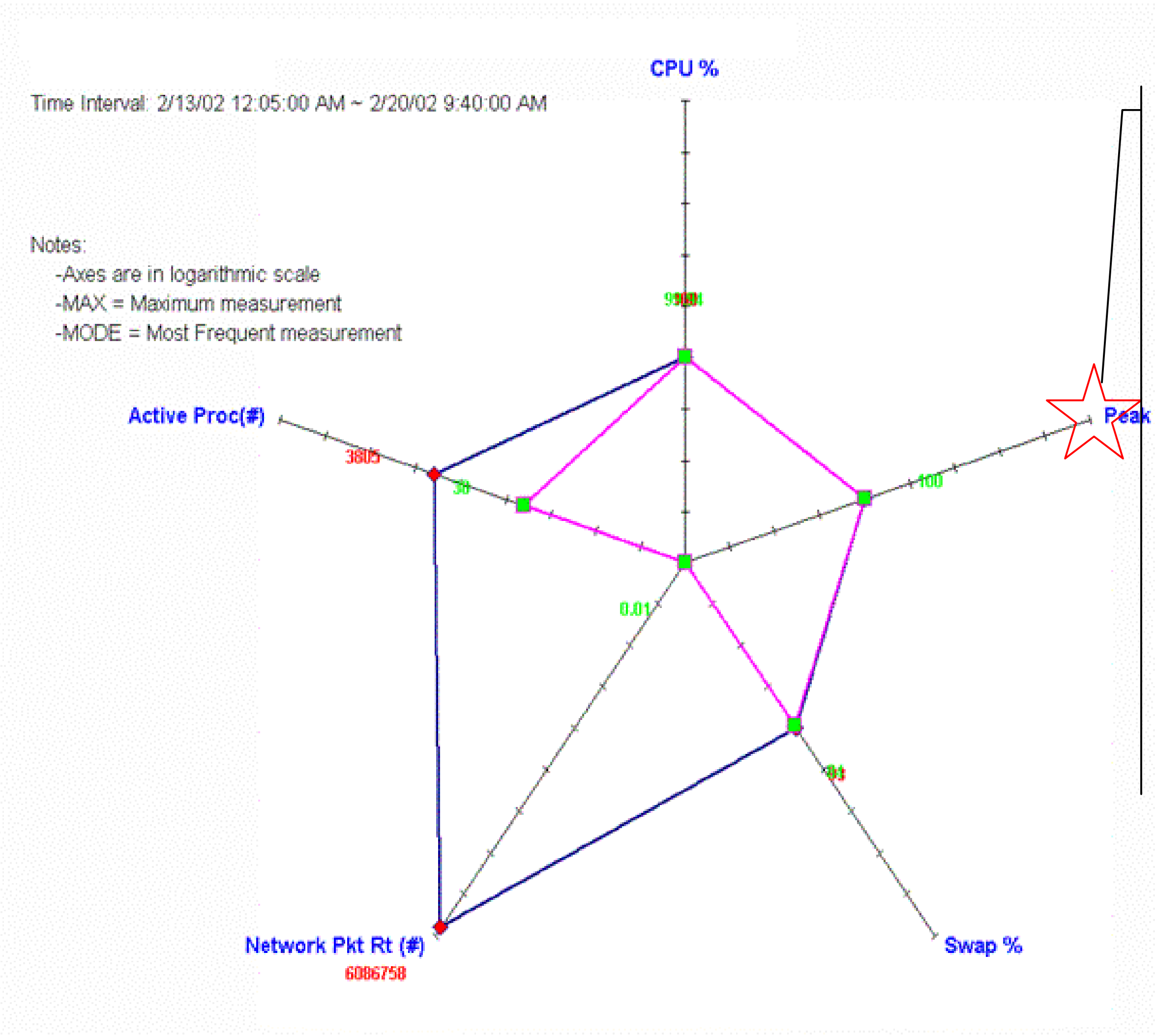
•The field recommends that Total CPU Utilization is between 40-60%. If a system regularly exceeds 60% utilization, they recommend considering a hardware upgrade.

•Many customers sampled do experience 100% Total CPU Utilization periodically, but the average and modal values on these systems are significantly lower than 100%.

•CPU System Mode Utilization seldom is greater than CPU User Mode Utilization on customer systems.

•Hardware problems are the only situations on customer systems where CPU System Mode Utilization exceeds CPU User Mode Utilization.

Stress Test Disk Utilization



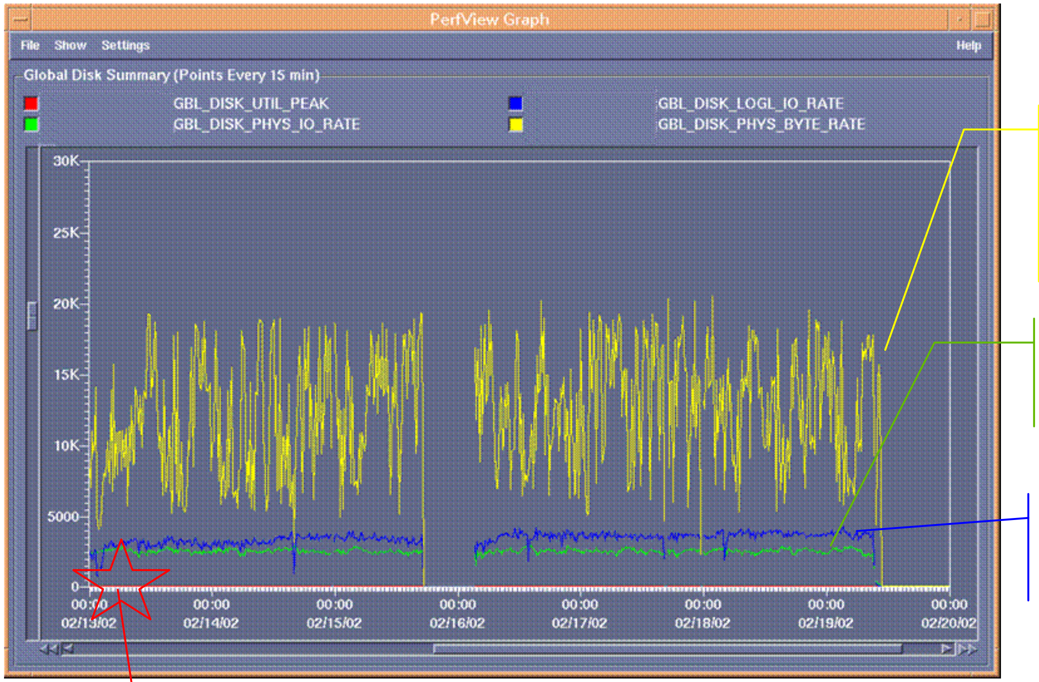
Time Interval: 2/13/02 12:05:00 AM ~ 2/20/02 9:40:00 AM

Notes:
-Axes are in logarithmic scale
-MAX = Maximum measurement
-MODE = Most Frequent measurement

Peak disk is percentage of time during the interval that the busiest disk device had IO in progress from the point of view of the Operating System. It is not an average utilization over all the disk devices. A peak disk utilization of more than 50 percent often indicates a disk IO subsystem bottleneck situation. A bottleneck may not be in the physical disk drive itself, but elsewhere in the IO path.

For this run, Max and Modal Disk Physical Byte Rate labeled as Peak Disk are both 100%.

Stress Test Disk Utilization



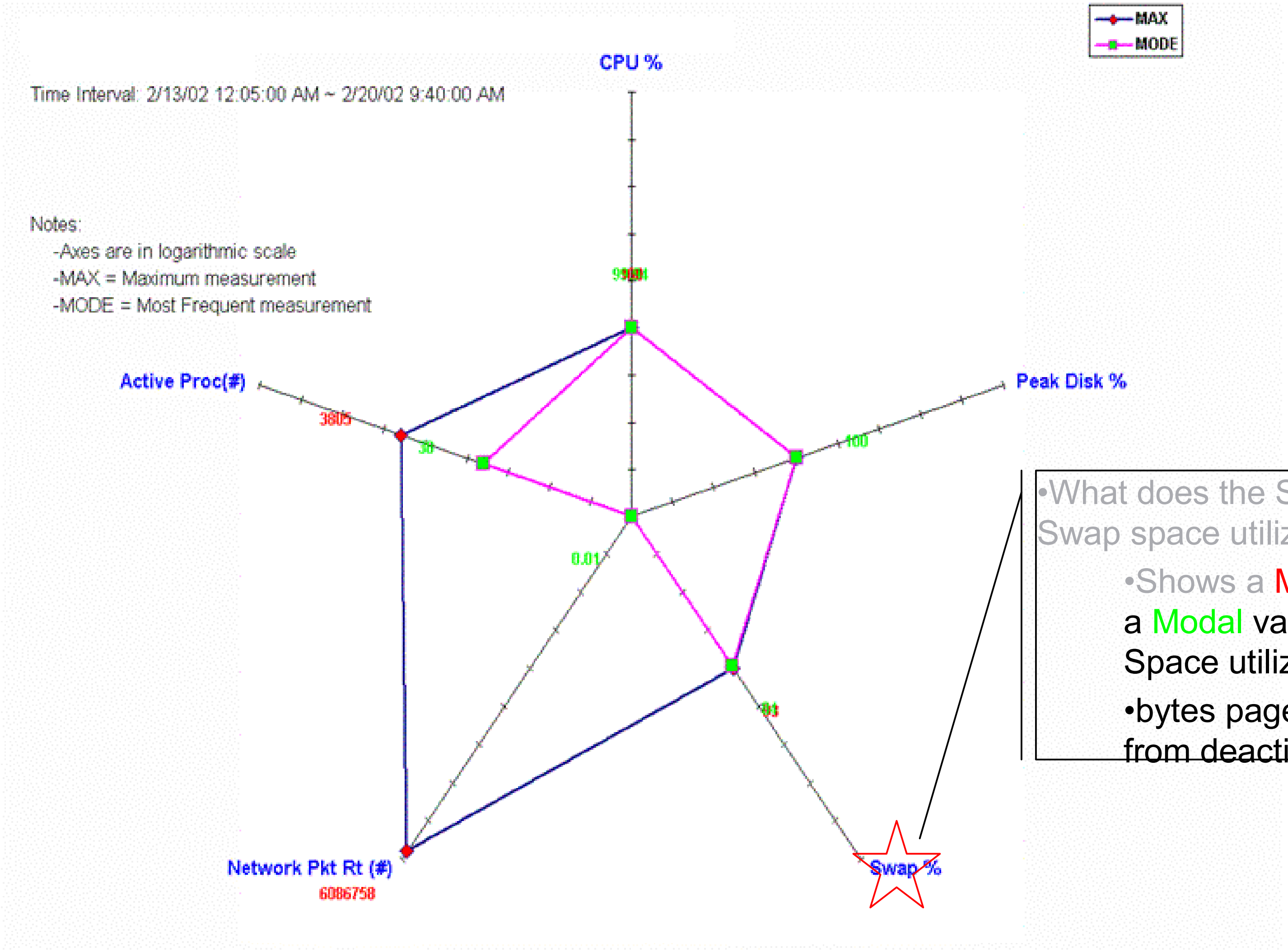
- Disk Physical Byte Rate variability is high and ranges from ~25K bytes to ~5K bytes.

- Disk Physical IO Rate is fairly constant at ~2500bytes

- Disk Logical IO Rate is fairly constant at ~4000%.

- Global Disk Util Peak is hard to see, but is constant at 100%. This value appears on the star chart.
- Customer Comparison: 30% of the customers we surveyed experienced max 100%. 20% experienced modal values 100%. Most customers has significantly lower max and modal values.

Stress Test Swap Utilization



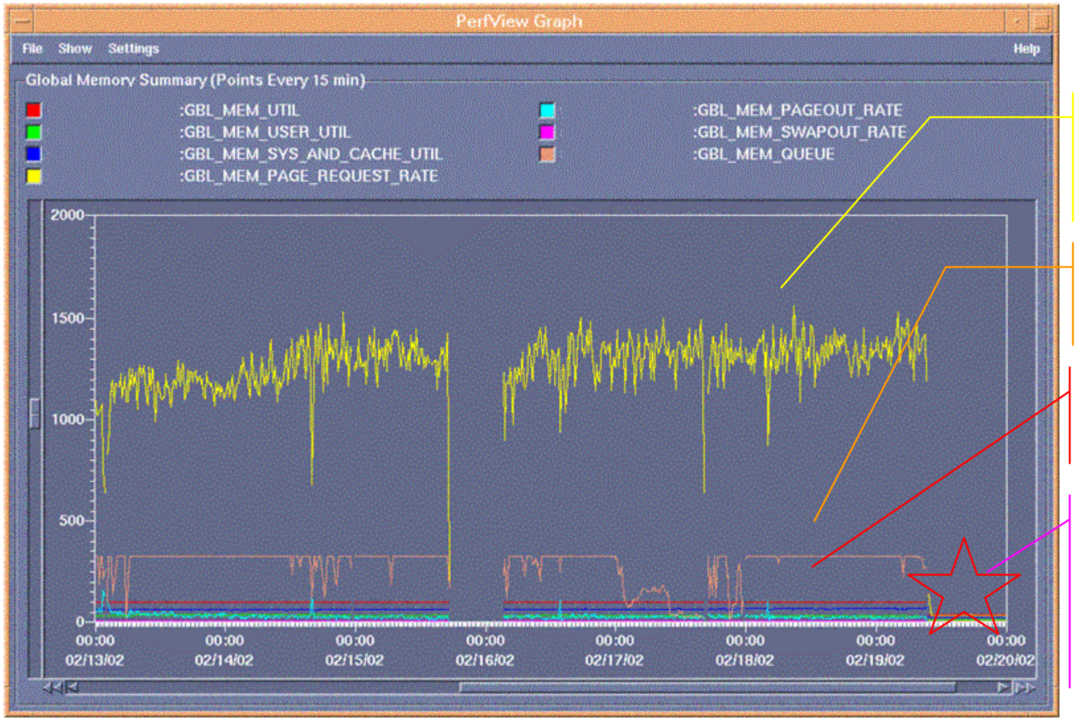
Time Interval: 2/13/02 12:05:00 AM ~ 2/20/02 9:40:00 AM

Notes:
- -Axes are in logarithmic scale
- -MAX = Maximum measurement
- -MODE = Most Frequent measurement

•What does the Star Chart tell us about Swap space utilization?

- •Shows a Max value of 94% and a Modal value of 84% for Swap Space utilization.
- •bytes paged out to swap areas from deactivated regions

Stress Test Swap Utilization



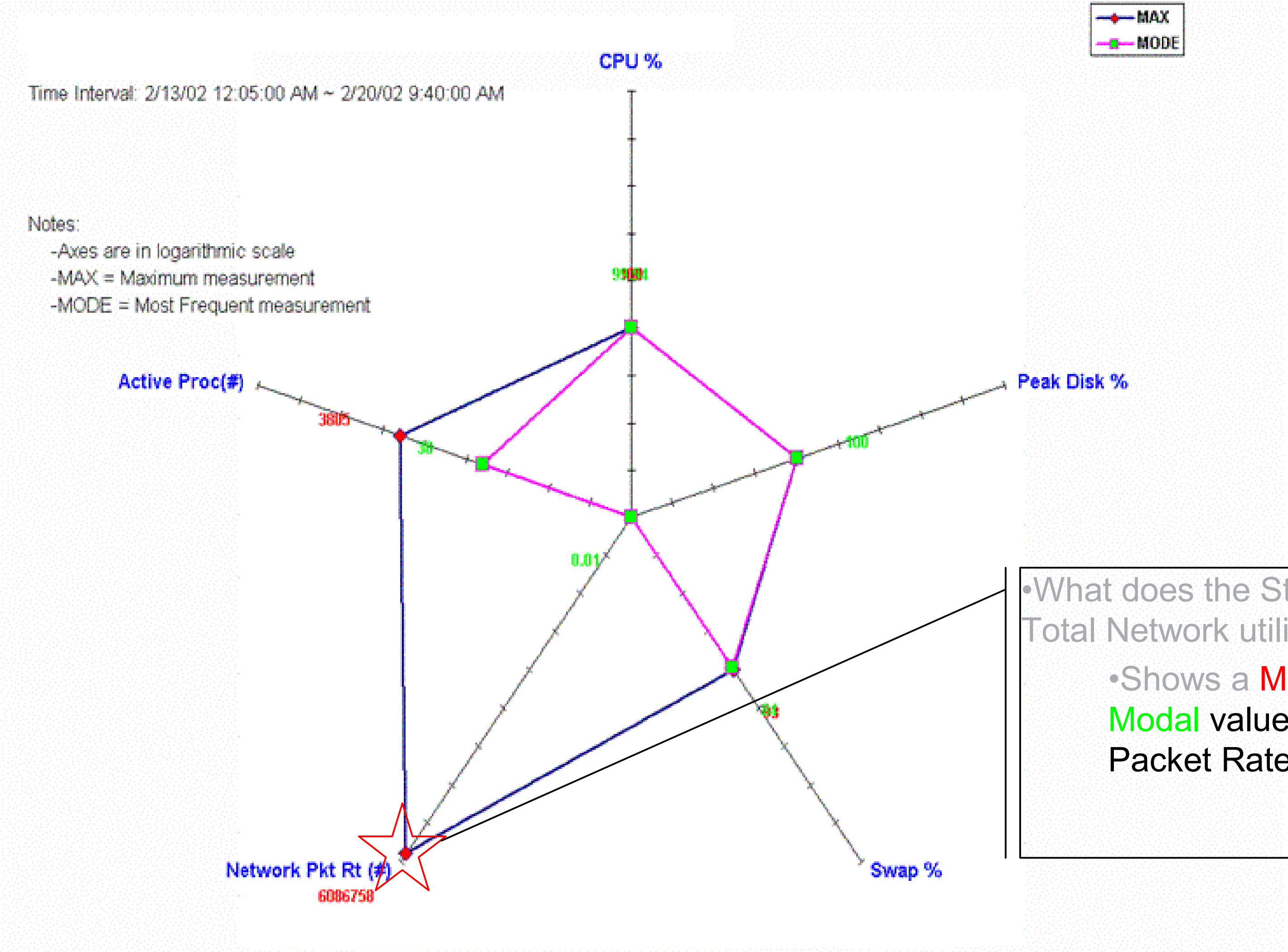
- Memory Page Request Rate variability is moderate mostly between 1000 and 1500.

- Memory Queue tends to be twice that of output packet rate.

- Memory Utilization is constant at 100%.

- Swap space utilization does not show up well on the memory line graph since the non percentage memory statistics influence the graph scale.

Stress Test Network Packet Rate

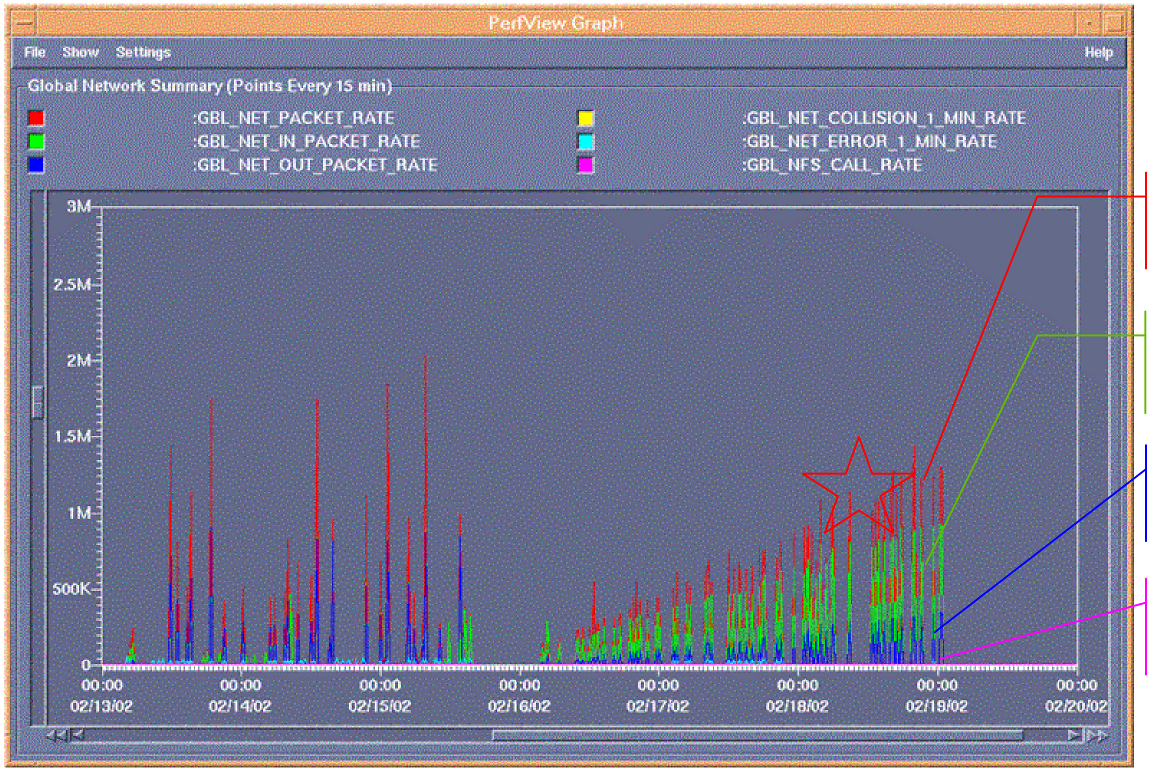


Time Interval: 2/13/02 12:05:00 AM ~ 2/20/02 9:40:00 AM

Notes:
-Axes are in logarithmic scale
-MAX = Maximum measurement
-MODE = Most Frequent measurement

- What does the Star Chart tell us about Total Network utilization?
  - Shows a Max value > 6MB and a Modal value of 0 for Total Network Packet Rate.

Stress Test Network Packet Rate

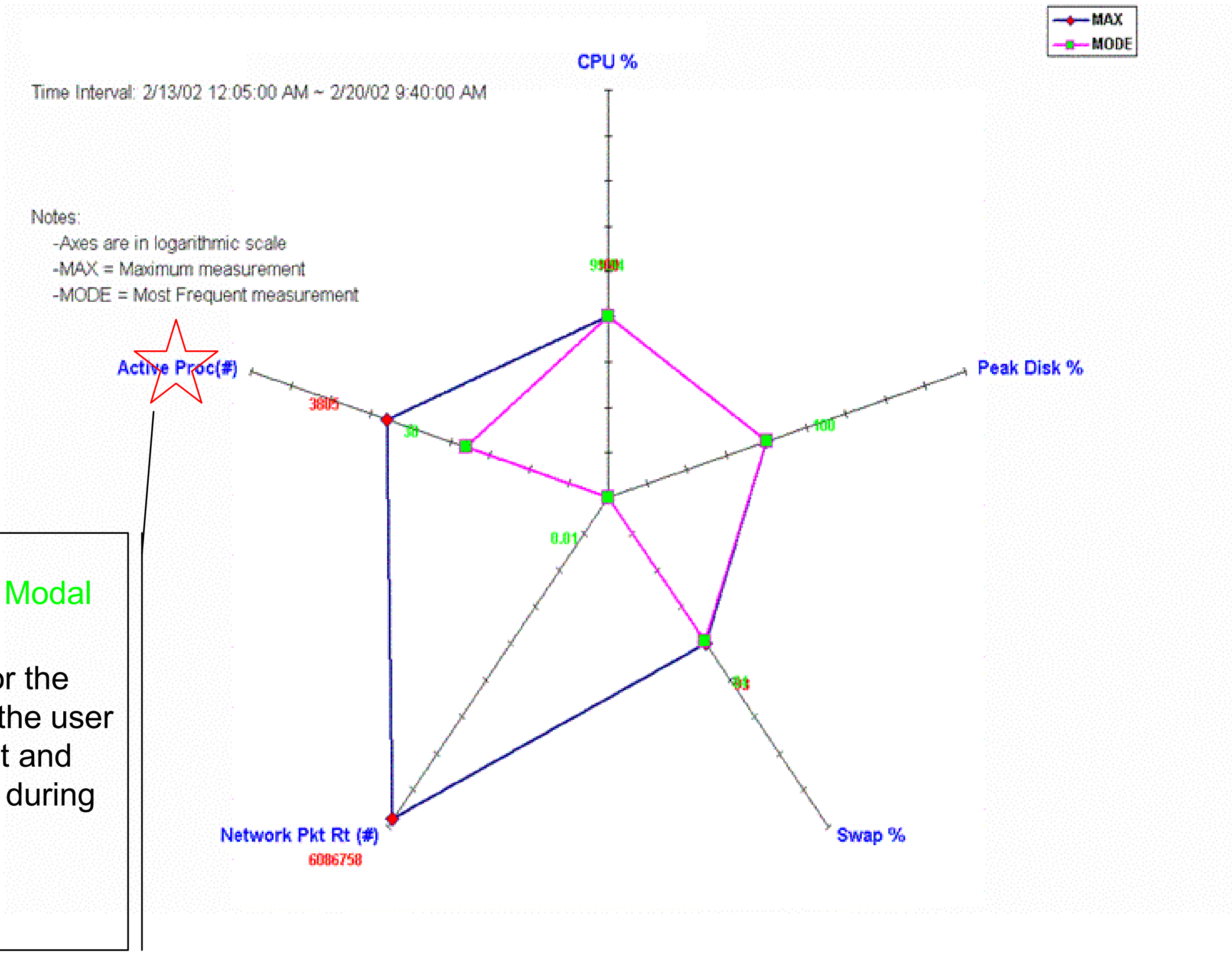


Global Network Summary (Points Every 15 min)

🟥	:GBL_NET_PACKET_RATE	🟨	:GBL_NET_COLLISION_1_MIN_RATE
🟩	:GBL_NET_IN_PACKET_RATE	🟦	:GBL_NET_ERROR_1_MIN_RATE
🟦	:GBL_NET_OUT_PACKET_RATE	🟪	:GBL_NFS_CALL_RATE

- Total Network Packet Rate variability is high ranging from 0 to 6MB.
- Network Input Packet Rate tends to be twice that of output packet rate.
- Network Output Packet Rate
- NFS Call Rate indicates no NFS activity.

Stress Test # Active Processes



Time Interval: 2/13/02 12:05:00 AM ~ 2/20/02 9:40:00 AM

Notes:

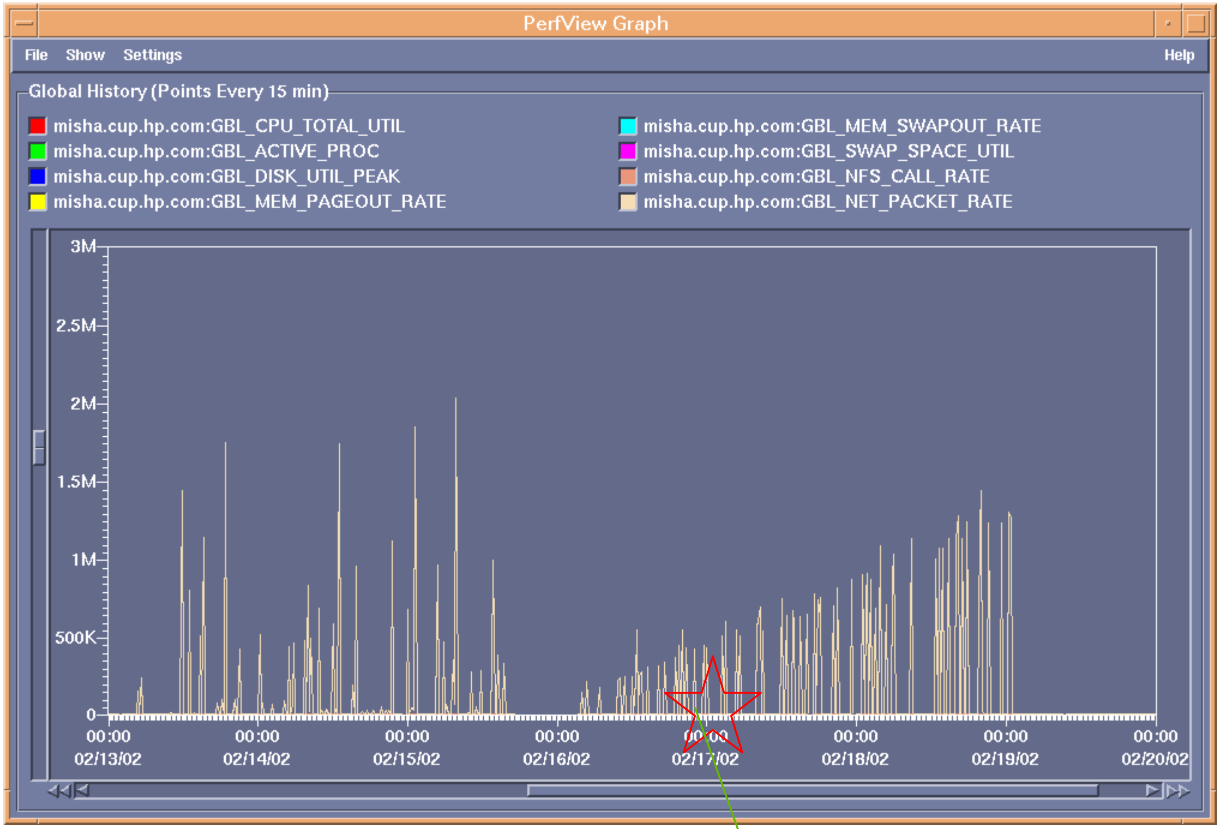
-Axes are in logarithmic scale

-MAX = Maximum measurement

-MODE = Most Frequent measurement

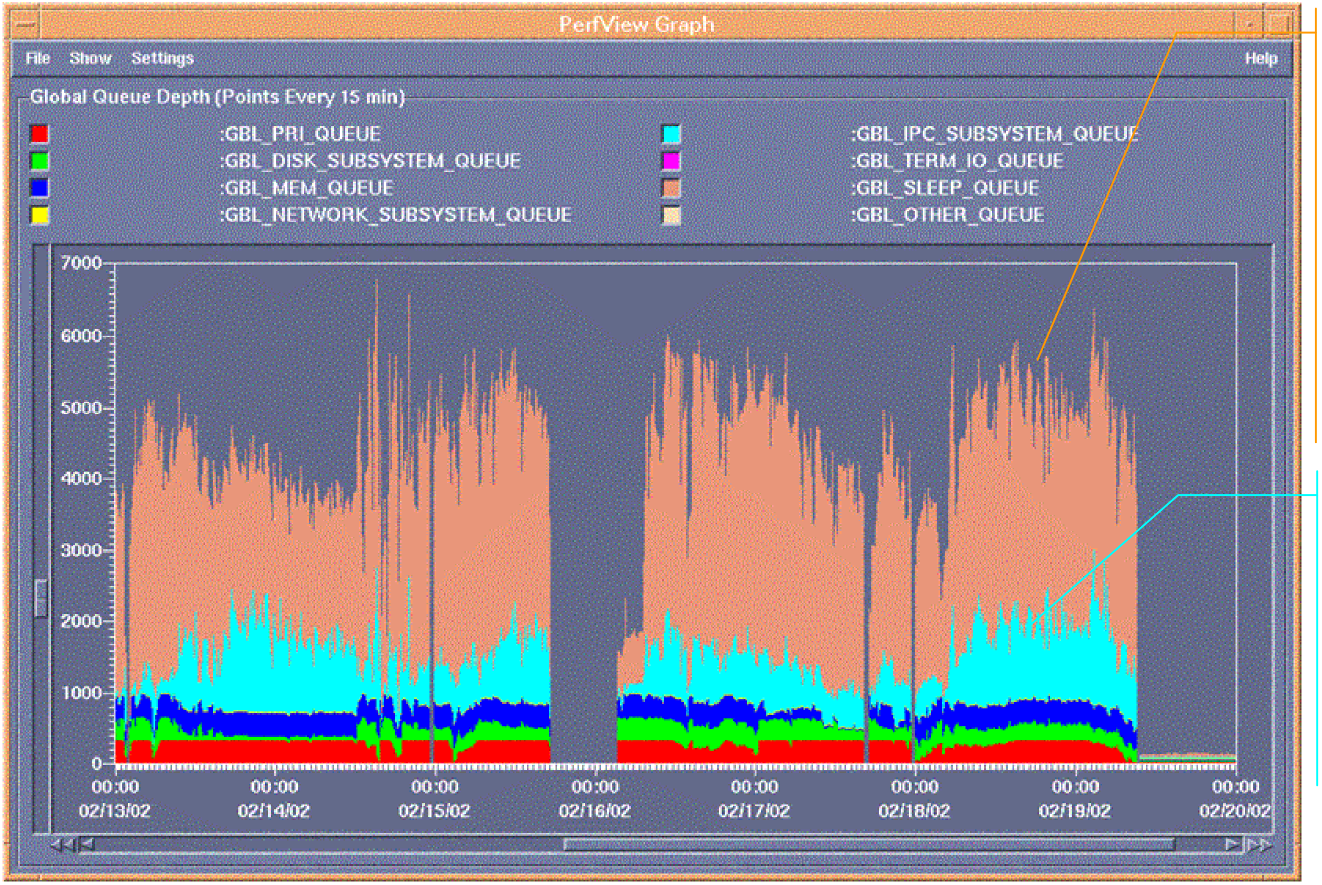
- For this run, Max Active Processes were 3805 the Modal value was 38.
- This value was chosen for the star chart because it tells the user how many processes exist and consume some CPU time during the interval

Stress Test # Active Processes


PerfView Graph
File
Show
Settings
Help
Global History (Points Every 15 min)
🟥 misha.cup.hp.com:GBL_CPU_TOTAL_UTIL
🟩 misha.cup.hp.com:GBL_ACTIVE_PROC
🟦 misha.cup.hp.com:GBL_DISK_UTIL_PEAK
🟨 misha.cup.hp.com:GBL_MEM_PAGEOUT_RATE
🟦 misha.cup.hp.com:GBL_MEM_SWAPOUT_RATE
🟪 misha.cup.hp.com:GBL_SWAP_SPACE_UTIL
🟧 misha.cup.hp.com:GBL_NFS_CALL_RATE
🟫 misha.cup.hp.com:GBL_NET_PACKET_RATE
3M
2.5M
2M
1.5M
1M
500K
0
00:00 02/13/02
00:00 02/14/02
00:00 02/15/02
00:00 02/16/02
00:00 02/17/02
00:00 02/18/02
00:00 02/19/02
00:00 02/20/02

•Active Processes are obscured on the Global History Graph due to high network traffic levels.

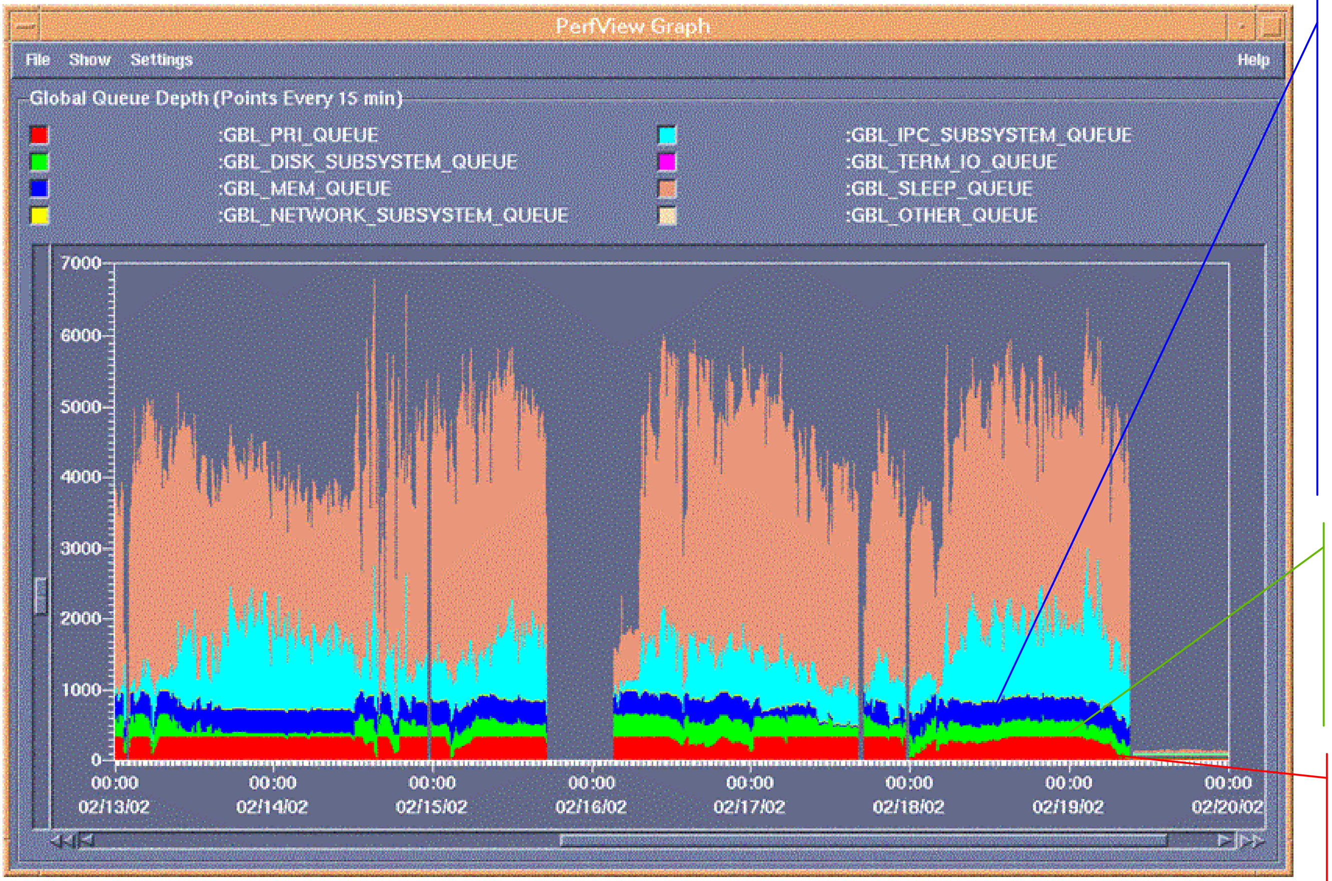
Stress Test Queue Utilization



- Sleep Queue is the average number of processes or threads waiting to awaken from sleep system calls during the interval, ie. waiting on system calls such as sleep, wait, pause, sigpause, sigsuspend, poll and select.

- IPC Queue is the average number of processes or threads waiting for IPC, MSG, SEM, PIPE, sockets) and streams IO wait states activity to complete during the interval.

Stress Test Queue Utilization

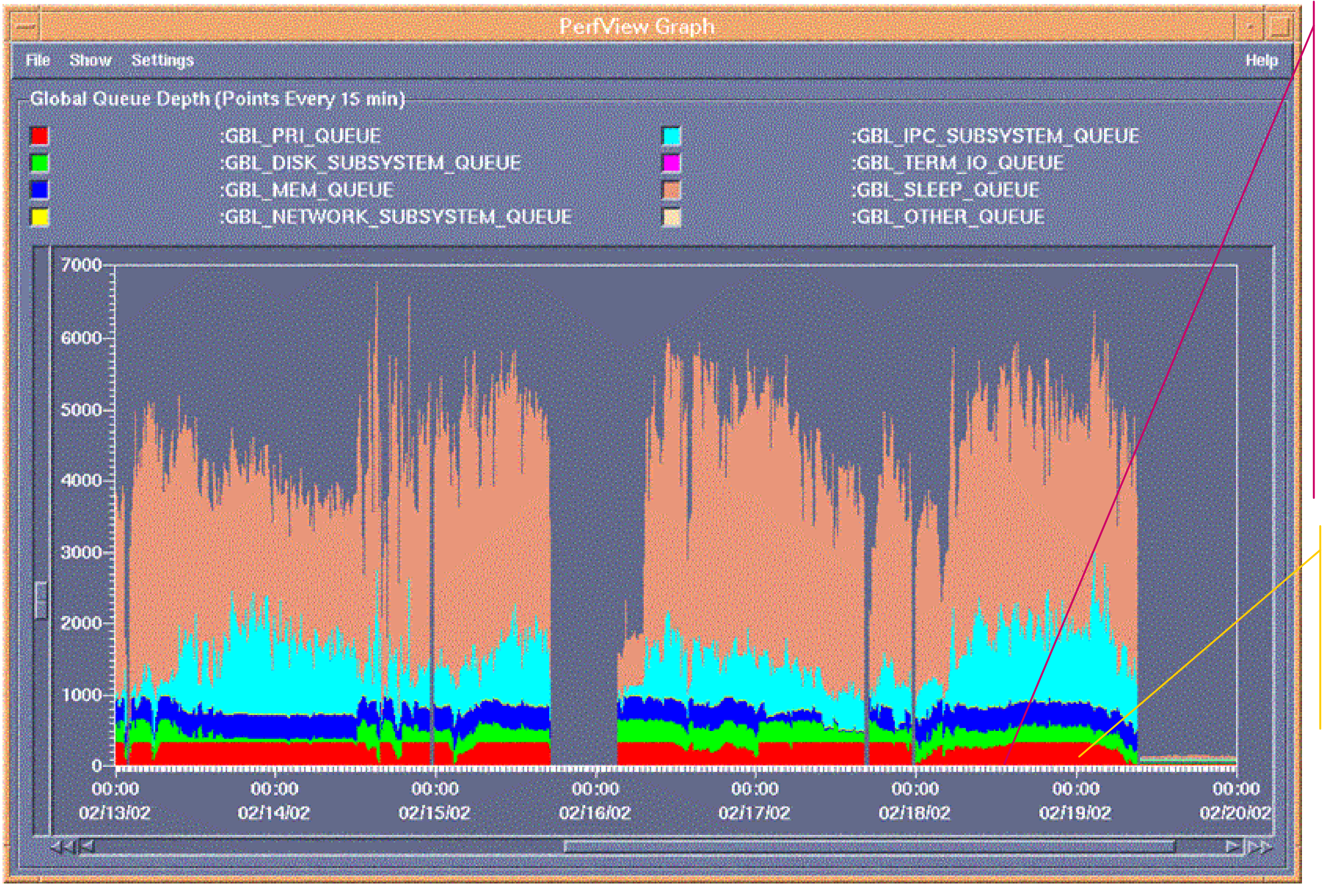


- Memory Queue is the average number of processes or threads waiting for virtual memory disk accesses to complete. This typically happens with large memory allocations or with heavy swapping. This is the first time we have seen significant memory queue activity.

- Disk Queue is he average number of processes or threads waiting for their file system disk IO to complete.

- Priority Queue is the average number of processes or threads waiting for their priority to become high enough to get the CPU during the interval.

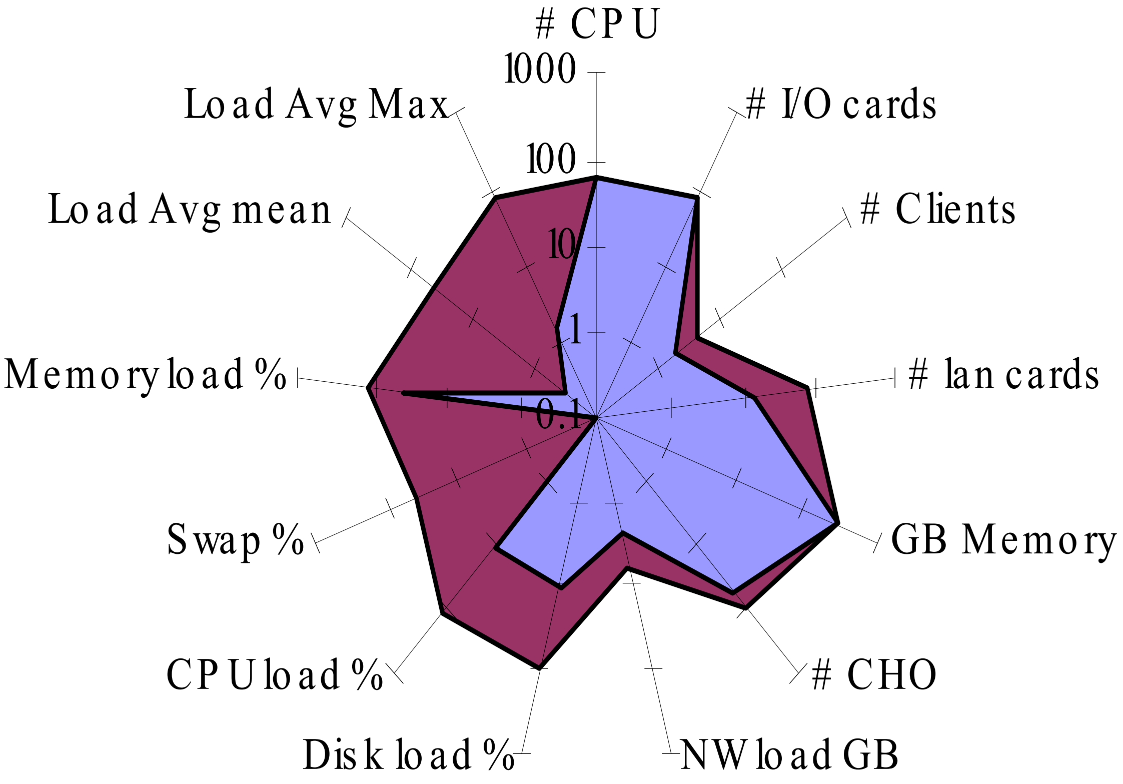
Stress Test Queue Utilization



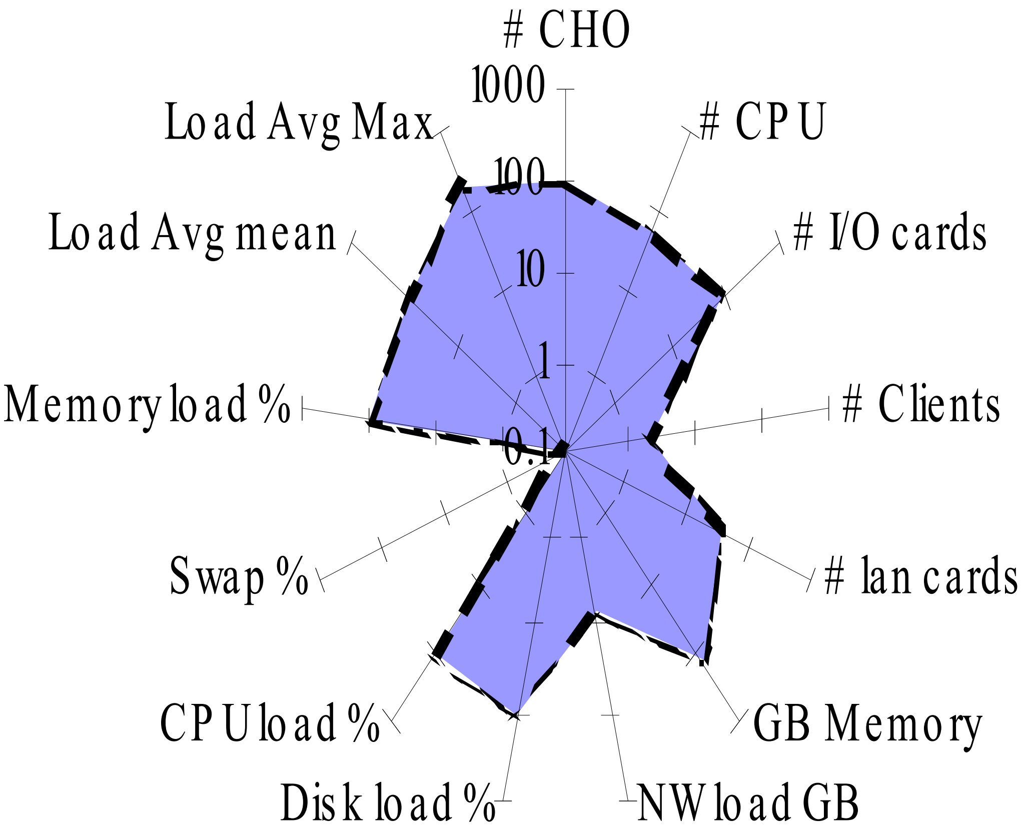
•Term IO Queue is the average number of processes or threads waiting for terminal accesses to complete. We have seen significant term io queue activity.

•Global Other Queue is sum of other queue activities. We have never seen other queue activity.

Stress Test Target vs. Attained: 2



Stress Test Target vs. Attained : Final



invent