
hp
invent

Presenter

Crisis Management Team Performance Troubleshooting

Ken Johnson
Escalation Engineer

Hewlett-Packard Company
100 Mayfield Avenue, MS 37UM
Mountain View, CA 94043

E-mail: ken_johnson@hp.com
Fax: (650) 691-3187

Purpose

Crisis Management Team Performance Troubleshooting

- To share the strategies and tactics used by the HP Crisis Management Team (CMT) to resolve performance escalations – using real world examples and case studies

- We will not deal with system tuning, capacity planning or benchmarking

Introduction

Crisis Management Team Performance Troubleshooting

1. The CMT Perspective: Emergency Room
2. Initial Steps in a Performance Escalation
3. Is the Work Necessary?
4. Looking for Anomalies
5. Isolating Components
6. Knowing Your System
7. Rules of Thumb

Module 1

The CMT Perspective: Emergency Room

- ER the TV show - we do triage to stop the bleeding
- Stabilize the system as fast as possible
- Quickly identify the first steps
  - Is this an HP defect/config issue / 3rd party issue ?
  - Often our value-add is to point in the right direction
- We have a system perspective
  - Understand interactions between HW, OS, Network, DB, Application
  - We train our engineers for a system perspective

Module 2

Initial Steps in a Performance Escalation

- Defining the Performance Problem
- Metrics – Know What the Thermometers are Measuring
- First Metric: System/User CPU Ratio
- Is There a Bottleneck?

Module 2

Defining the Performance Problem

Things we want to know in the first minutes

- When did the performance problem start?
- How do you know you have a problem ?
  - Is this a user/business impacting problem or a metric-only issue ?
  - Is the problem quantified ?
- Is the hardware and OS base stable and consistent?
- What changed ?

Module 2

Defining the Performance Problem

- Quantification
  - Allows you to measure the objective effect of changes
  - Define the current state and the goal
- Changing only one thing at a time
- Characterize the problem
  - System wide or particular application?
  - All the time or specific time of day?
  - Network access or local access?
  - NFS mounts or local disks?
  - Consistent or erratic?

Module 2

Metrics – Know What the Thermometers are Measuring

- Metrics are simply statistics produced by software
  - Some of our escalations are with performance tools
- Be sure what a metric is really measuring
  - Wait time/service time
  - Page out/swap out
  - Run queue/load average
  - Inode table utilization
- Always have more than one data point and always use more than one tool
- Your tools can affect the environment

Module 2

First Metric: System / User CPU Ratio

- What is system CPU?
- Why is it important?
  - Points at initial directions to pursue root cause
  - HP owns this code
- High system CPU can point to:
  - High number of system calls
  - Memory/IO problems
  - Thrashing/spinning in the kernel
- CMT has visibility into system CPU utilization
  - There are utilities we use to do kernel profiling on production systems

Module 2

Is There a Bottleneck?

- This is the supply side of performance
- Easiest to look at – easiest to fix

- I/O
  - Is there queuing on any drives?
  - Are there long service/wait times on any drives?
- CPU
  - Is there a significant load average?
  - Is system CPU high?
  - Are processes priority waited?
- Memory
  - Is there any paging or deactivations?
  - Is there significant swap utilization?

Module 3

Is the Work Necessary?

- Is the IO Demand Efficient?
- Are the CPU Cycles Necessary?
- Is the Application Efficient?
- Is the Memory Utilization Necessary?

Module 3

Is the IO Demand Efficient?

System:	K580 4-way 11.0	Manufacturing
	2 GB memory	
	Database server	
Symptoms:	2 year installation	
	Suddenly batch jobs taking much longer to execute	
	No changes to programs or database settings	
	No system bottlenecks	
	Elevated IO rate but no queuing and fast service times	
Diagnosis:	Cost based plan had been used for key queries in DB	
	Several of the queries started doing serial IO	

Module 3

Are the CPU Cycles Necessary?

System:	T600 8-way 10.20 2 GB memory Development system Compiling and source code management
Symptoms:	3 year installation Recently seeing slow overall performance Intermittent High system CPU and high context switch rates
Diagnosis:	Files used for compiling were located in one directory Large number of files and very volatile Contention around the directory file itself (25 MB) Spinning while waiting for shared resource caused unnecessary context switching

Module 3

Is the Application Efficient?

System:	N4000 4-way 11.0 4 GB memory Web server
Symptoms:	New installation Server throughput was never acceptable High CPU utilization with mostly user CPU Load average was reasonable and good system response time
Diagnosis:	Identified large # of semop calls in bolt-on application Application was in the critical path for the server Allowed vendor to identify configuration problem

Module 3

Is the Memory Utilization Necessary?

System:	V2500 16-way 11.0 4 GB memory Database server
Symptoms:	Memory utilization at 100% High page out and deactivation rates
Diagnosis:	Default 50% buffer cache had been used Maxuser had been set very high – affects many other kernel variables Final solution was to add memory and to tune kernel variables

Module 4

Looking for Anomalies

- System Call Rates/CPU Utilization
- IO Patterns
  - By Device
  - By Time of Day
  - By Process
- Wait States
  - Global and Per Process

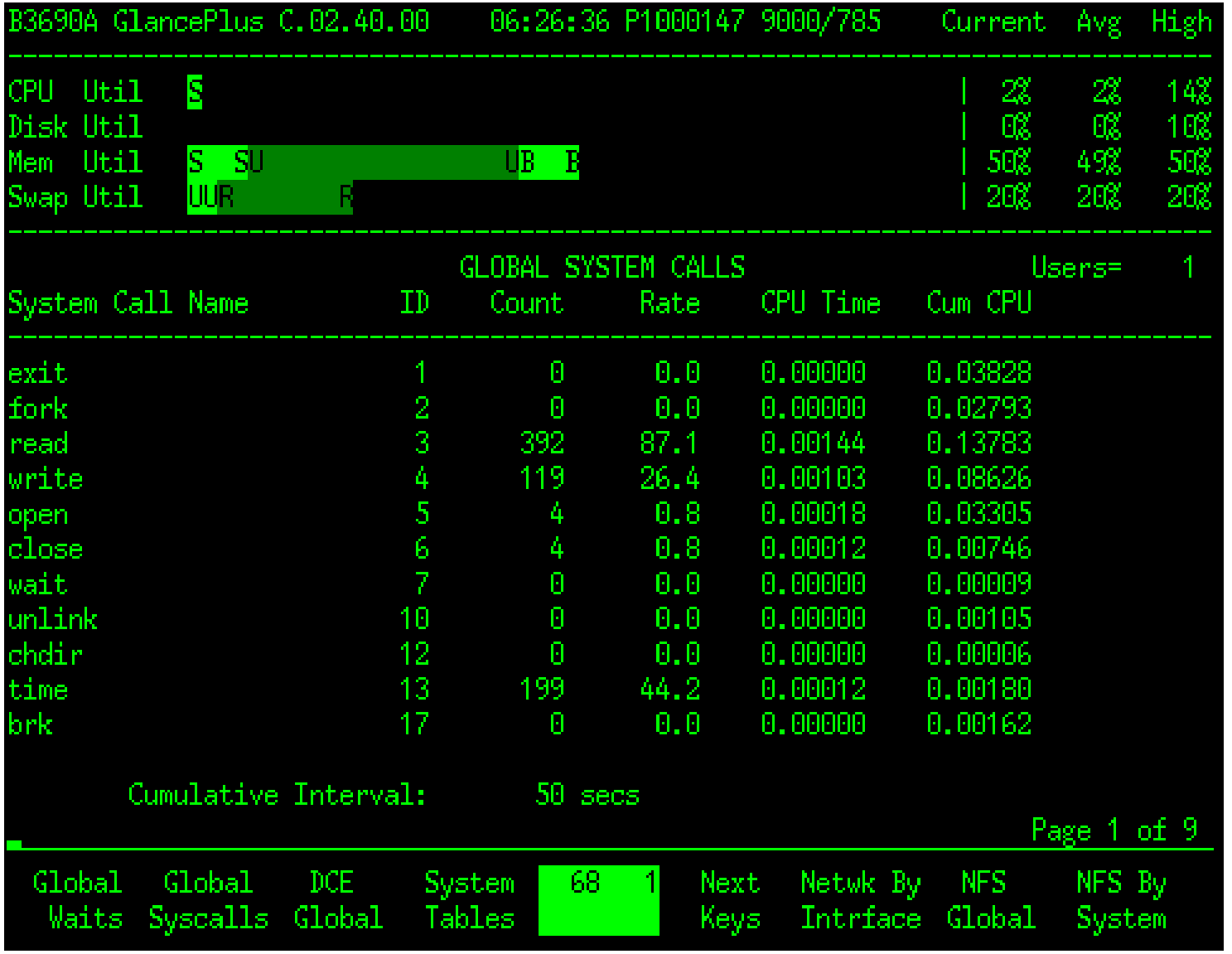
Module 4

System Call Rates/CPU Utilization

System: K460 4-way 10.20
2 GB memory
Legacy shellscript-based application
Files ftp'd in, processed, then put in a directory for pickup

Symptoms: Suddenly application throughput was down
No changes to the application
System CPU way up

Module 4



Module 4

System Call Rates/CPU Utilization

System:	K460 4-way 10.20 2 GB memory Legacy shellscript-based application Files ftp'd in, processed, then put in a directory for pickup
Symptoms:	Suddenly application throughput was down No changes to the application System CPU way up
Diagnosis:	vfork() was very large CPU consumer Identified shellscript that was in a 'bop'

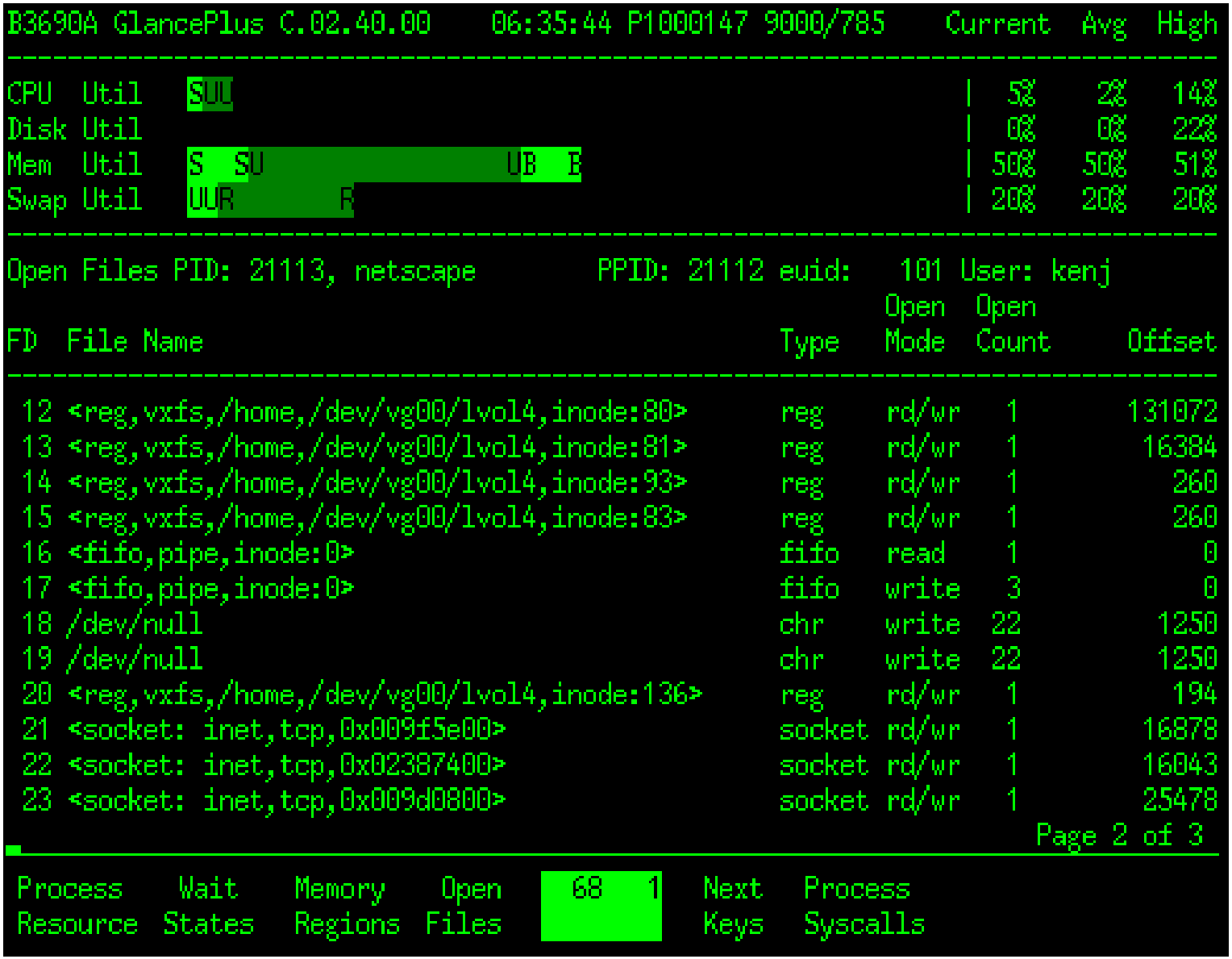
Module 4

IO Patterns - By Device, By Time of Day, By Process

System: N4000 4-way 11.0
4 GB memory
Database server for web front-end

Symptoms: New installation
System response was good
Unacceptable database performance
DB connections were short-lived
Analysis showed that delay was in DB disconnet

Module 4



Module 4

IO Patterns - By Device, By Time of Day, By Process

System: N4000 4-way 11.0
4 GB memory
Database server for web front-end

Symptoms: New installation
System response was good
Unacceptable database performance
DB connections were short-lived
Analysis showed delay was in DB disconnect

Diagnosis: Used Glance to observe when user disconnected
Found high rates of IO during disconnect
IO was to 2 database trace files

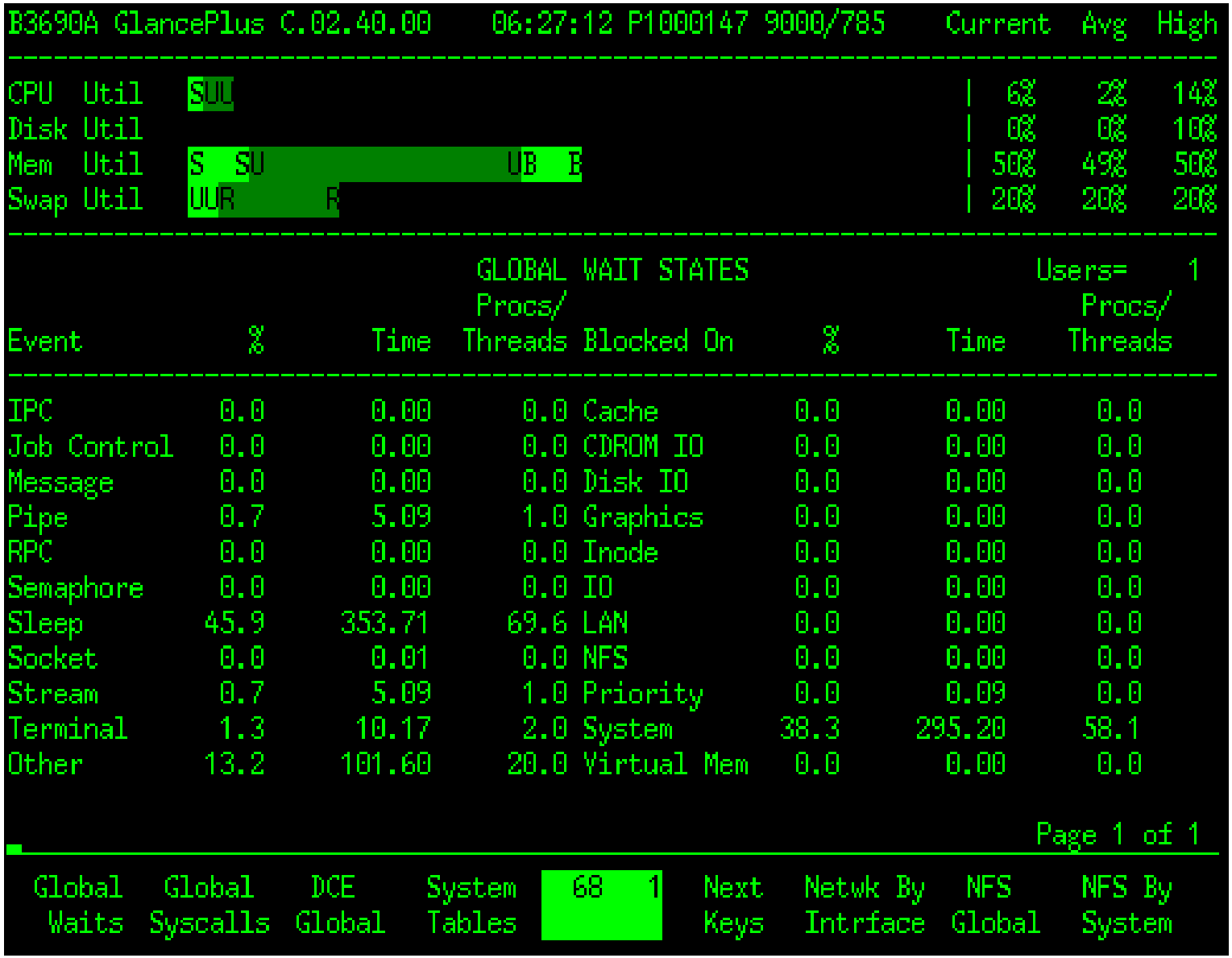
Module 4

Wait States – Both Global and Per Process

System: V2250 8-way 11.0
8 GB memory
Database server

Symptom: New installation
Slow database throughput
No system bottlenecks or high utilization

Module 4



Module 4

Wait States – Both Global and Per Process

System: V2250 8-way 11.0
8 GB memory
Database server

Symptom: New installation
Slow database throughput
No system bottlenecks or high utilization

Diagnosis: Identified high semop waits
Database tuning required

Module 5

Isolating Components

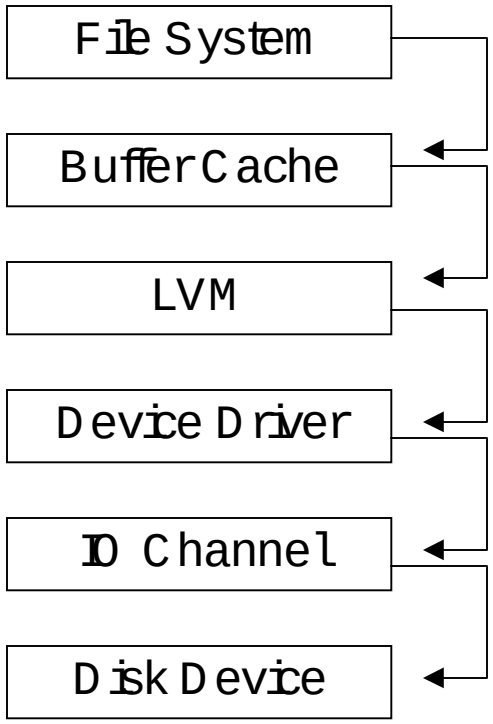
Make everything into a black box

Define and manipulate inputs and outputs

- The Discrete Elements of an IO Request
- Taking the Network Out of the Picture
- OmniBack Performance Debugging Techniques

Module 5

The Discrete Elements of an IO Request



mincache = direct

/dev/vg/rlvol

/dev/rdsk/cxtxdx

Module 5

The Discrete Elements of an IO Request

- Bottlenecks can happen at any of the layers in either direction
- Isolate the IO test at one layer
  - mincache = direct
  - /dev/vg/rlvol
  - /dev/rdsk/cxtxdx
- Only test reading or writing

Module 5

The Discrete Elements of an IO Request

Code Fragment for Timing IO Requests

```
#include <sys/time.h>

#define delta_tv(tv_0, tv_1)\
   (tv_1.tv_sec - tv_0.tv_sec + (tv_1.tv_usec - tv_0.tv_usec)/1000000.0)

struct timeval xtv0, xtv1;
struct timezone tz;
double rdt= 0.0

main()
{
   gettimeofday(&xtv0, &tz);
   read(fd,buf,bufsize)
   gettimeofday(&xtv1, &tz);

   rdt= delta_tv(xtv0, xtv1);

   printf("milliseconds for read: %.3lfms)\n", 1000*rdt);
}
```

Module 5

Taking the Network Out of the Picture

- Multi-tiered applications (e.g. SAP) have large network components which can have a large impact on overall throughput
- Database access is often through sockets
- Techniques for isolation
  - Make local queries rather than client queries
  - With system issues execute problematic commands at the console
  - Use programs/benchmarks similar to those used for IO testing

Module 5

OmniBack Debugging Techniques

- Understand the capabilities of each component in the configuration
  - Examples:
    - T5xx IO backplane overwhelmed by more than one FW SCSI card
    - Modern DLTs need dedicated FW SCSI card/several disk-readers
- Isolate
  - Disk IO
  - Network
  - Tape IO
  - Updates to OmniBack database
  - Data compressibility

Module 5

OmniBack Debugging Techniques

Measuring Data Compression

```
cat file | compress -v > /dev/null
```

```
$ cat dedbg | compress -v > /dev/null
  Compression: 82.43%
$ cat dedbg | compress -v > dedbg.Z
  Compression: 82.43%
```

Module 6

Knowing Your System

- Transaction reporting
  - Example: SAP instrumentation
    ARM instrumentation
- Maintain a history
  - sar, vmstat, scope, application measures
- Develop an intuition for your systems
- Watch it closely when its healthy
- Know the performance pattern over the day/week/month
- Internals knowledge of the application/database
- Internals knowledge of the OS

Module 7

Rules of Thumb

- CPU
- MEMORY
- IO

Module 7

CPU Rules of Thumb

- System CPU <= 30%
- Total CPU < 80%
- Small bad average

Module 7

MEMORY Rules of Thumb

- Never page out
- Never deactivate processes
- Buffer cache < 500 MB

Module 7

I/O Rules of Thumb

- Utilization < 50% on any drive
- Minimal queuing < 4
- Response time ~10 milliseconds