# Taming the Terabytes – Case Study in Tuning a Data Warehouse.

**Sanjay Umarji**

Solution Architect

Oracle Solution Design Consulting, ESS

HP WORLD 2003

Solutions and Technology Conference & Expo

# Agenda

- Introduction and Background
- Data Warehouse Environment
- Pre-tuning behavior
- Tuning Exercise
  - Statspack, System Data, Event 10046 Trace, Wait Events
  - Data Observations, Analysis
  - Corrective actions
- Post-tuning behavior
- Conclusion

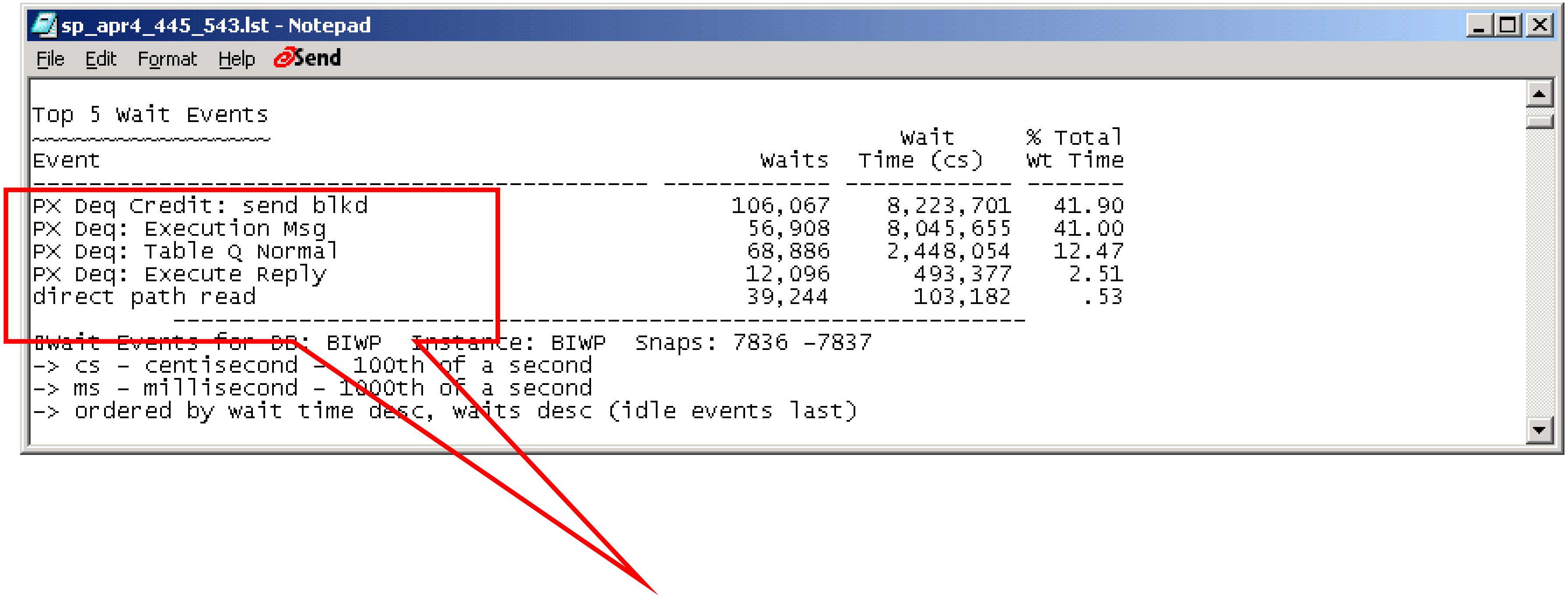
# Case Study

- A large Data warehouse over 4 TB
- GS160, 8 Cpu , 16 GB RAM
- Nightly Load Window of 12:00AM to 7:30AM
- New jobs related to Inventory data load added in the past few months.
- Load window extended into morning hours (8:45 AM) over last few months
- IT not able to meet the SLA to users

# Customer Observations

- Random spikes in performance of certain load jobs
- Continuous degradation in performance of large load jobs such as Inventory
- Informatica Jobs hanging for 2 hours before normal completion
- Swap space utilization on the rise

# Pre-visit Statspack

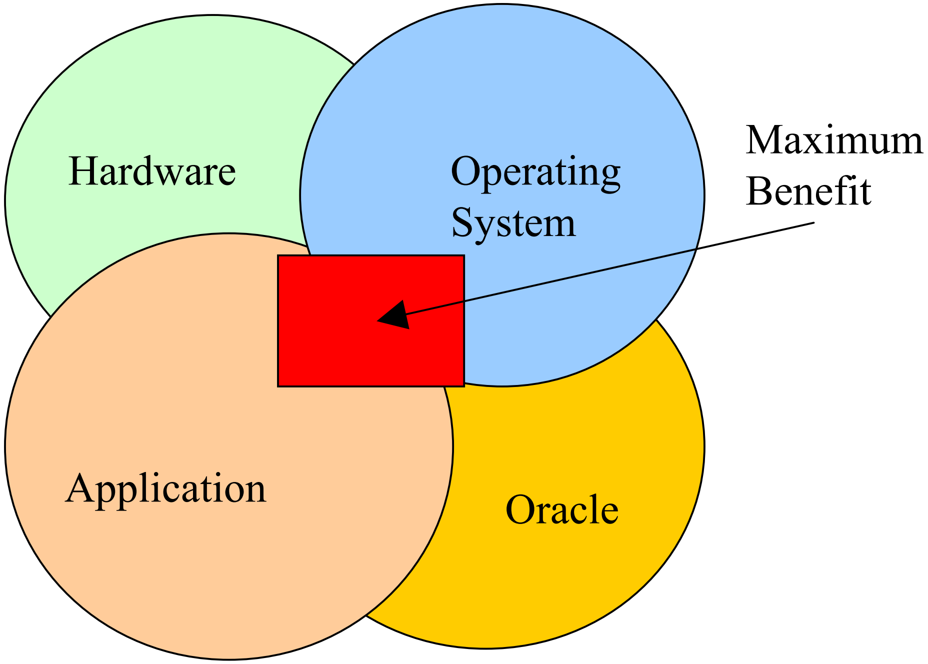


```
Top 5 Wait Events
~~~~~~~~~~~~~~~~~                                                    Wait     % Total
Event                                               Waits    Time (cs)   Wt Time
-------------------------------------------- ------------ ------------ -------
PX Deq Credit: send blkd                          106,067    8,223,701   41.90
PX Deq: Execution Msg                              56,908    8,045,655   41.00
PX Deq: Table Q Normal                             68,886    2,448,054   12.47
PX Deq: Execute Reply                              12,096      493,377    2.51
direct path read                                   39,244      103,182     .53
          -------------------------------------------------------------
Wait Events for DB: BIWP  Instance: BIWP  Snaps: 7836 -7837
-> cs - centisecond -  100th of a second
-> ms - millisecond - 1000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

- Most of these are generally idle Wait Events

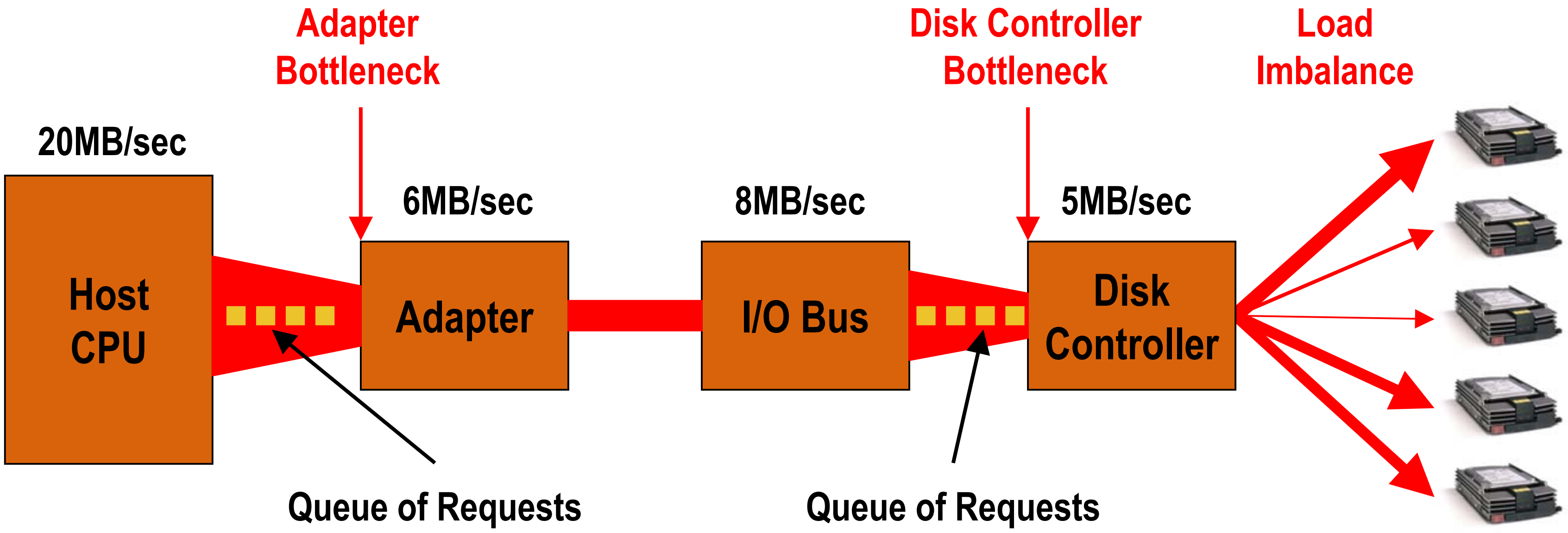
# System Performance

- One subsystem tuning
  - Yields one-sided solution
  - Could worsen performance of other subsystems
- Investigate performance of all subsystems
- Observe overlaps
- Narrow the focus to overlap
  - Yields maximum tuning benefit



# "There is always a Bottleneck"

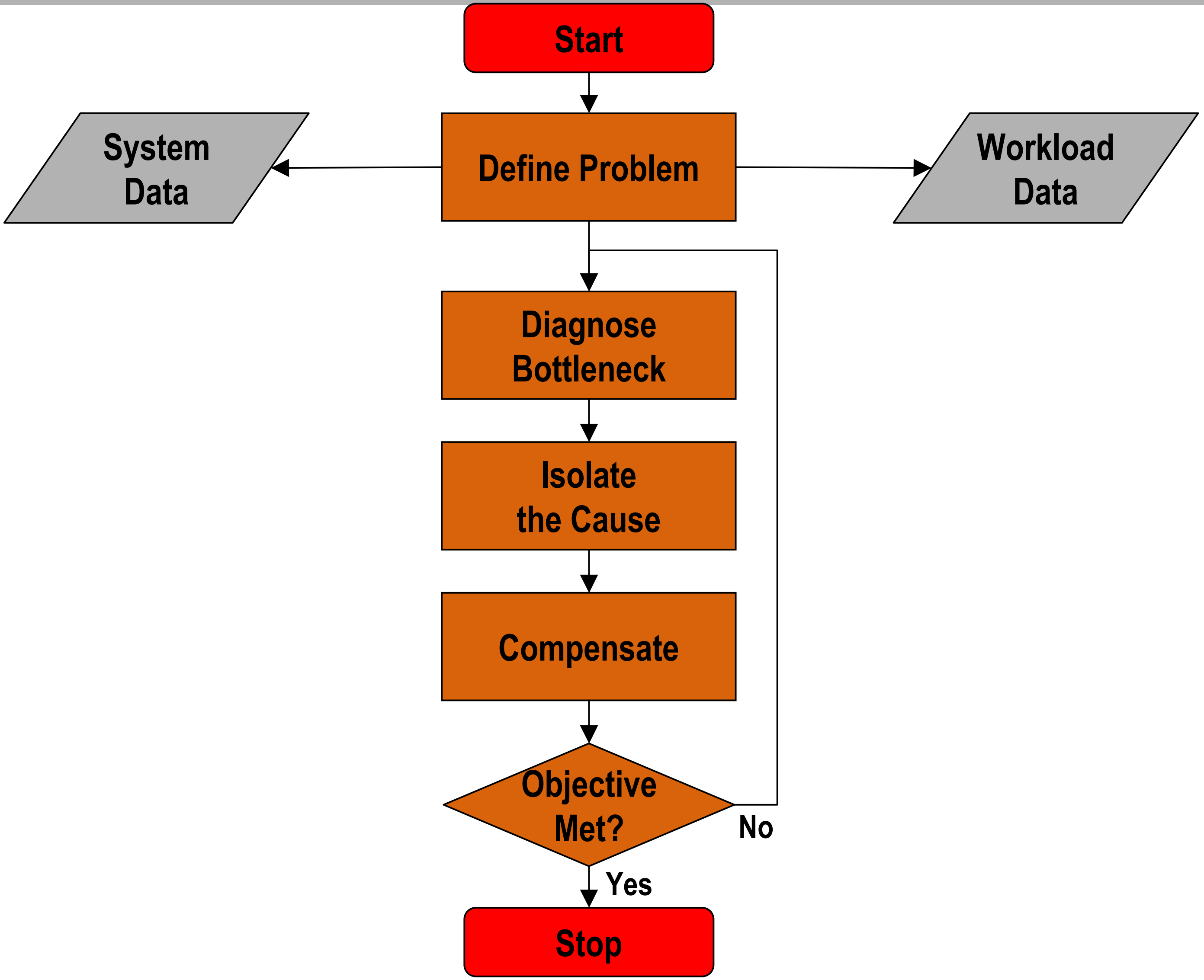
- Look for the weakest link
- Reconfiguration may solve problems



# Bottleneck (*continued*)

- Hardware
  - Disk subsystem
  - Processors
  - Memory
  - Network
- Operating system
  - Kernel Subsystems
- Database
  - Memory contention
  - Storage organization
  - I/O contention
  - Latch contention
  - Process contention
- Application
  - Efficiency of code
  - SQL
  - Indexes and locking

# Tuning Process



# Amdahl’s Law

$$S = \frac{1}{(1 - f) + \frac{f}{k}}$$

s = The effective Speedup

f = The Fraction of Work in Faster Mode

k = The speedup While in Faster Mode

# Amdahl’s Law

- **New Processor is five times faster (k=5)**
- **25% spent on I/O (f=0.75)**

$$S = \frac{1}{(1 - 0.75) + \frac{0.75}{5}} = 2.5$$

- New system is Only 2.5 times faster.

# Tools

- System Snap-shot Tools
  - What is happening at a given instant
  - Detailed Data

- Time Span Tools
  - Information over a time interval
  - Good for a trend identification

# Typical Observations from Collected Data

- Virtual memory statistics
  - Paging in and out of memory
  - Paging within memory
  - Working set size for a particular process
  - Amount of free memory
  - Page file size
- I/O statistics
  - Unbalanced load on disks
  - High disk service times
  - Long waiting queues
  - High disk utilization
  - Number of reads and writes per second
  - Read and write transfer rates for disks
  - Saturation of SCSI buses, host bus adapters, array controllers

# Typical Observations from Collected Data (*cont.*)

- Processor statistics
  - % processor utilization
  - Kernel time
  - User time
  - Processor queue length (Run Queue)
- Operating system statistics
  - Kernel waits/locks
  - File system statistics
- Oracle statistics
  - Cache hit ratios for Oracle
  - Rollback space utilization
  - Number of transactions committed and rolled back
  - Checkpoint and log switch frequencies

# Ratio-Based Tuning

- Ratios do not
  - Identify bottlenecks
  - Indicate why the performance is unsatisfactory
  - Expose the top bottlenecks in the system
  - Signal when to stop tuning
  - Expose whether the problem is within Oracle or outside
- Unanswered questions
  - My cache hit ratio is 99.999%. Why do I have performance issues?
  - Check all subsystems
  - Is there any hope?

# Wait Event Based Tuning

- Addresses these questions
  - What is my system waiting for? Where is the Pain?
  - What are the obstacles preventing higher performance?
  - Should I worry about the wait events and nothing else?
  - Why am I tuning disk I/O if there is no waiting on disk?
  - Why am I improving buffer cache hit ratio, if memory is not an issue?
  - What would you rather tell the management?

    Buffer cache hit ratio is only 95% and not improving, *or*

    Our bottleneck is one bad SQL statement and we are rewriting it.
- Provides
  - A better communication method for the DBA
  - A reliable and scientific way to precisely pinpoint root cause
  - Waits are Application Independent

# Oracle Wait Events

- First introduced in version 7 with little documentation
- Documentation improved in version 8 (Over 200 wait events)
- Oracle continues to add wait events (currently ~300 wait events in 9i)
- Examples
  - Log buffer space – process is waiting for free space in the log buffer
  - Database file sequential read – the process is waiting for a block to be read from the disk (indexed query)
- Wait events are collected by Oracle if
  - timed_statistics = 1 (init.ora) - Can be set dynamically
  - Timing Interval is Critical

# Oracle Wait Events (*continued*)

- Can be viewed using WAIT interface (v$ tables)
  - V$system_event
  - V$session_event
  - V$session_wait
  - V$event_name

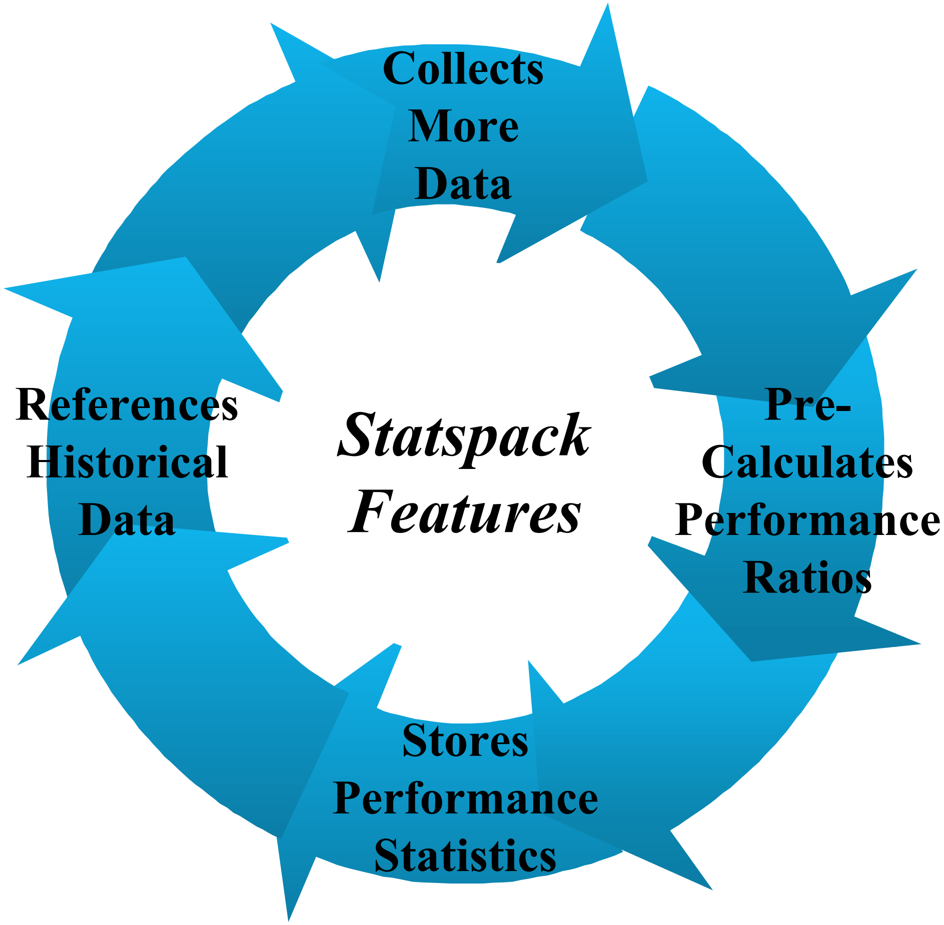
- Historical information
  - Not available in above tables
  - Must use a tool for snapshots and historical information

    a) BSTAT/ESTAT<br>
    b) StatsPack

# BSTAT/ESTAT

- Command-line interface that gathers instance performance data
- Consists of
  - UTLBSTAT.SQL
  - UTLESTAT.SQL
- Captures a single snapshot of performance data between specific start and end times
- Drawbacks
  - Reports are difficult to read and interpret
  - No latest Oracle features and functionality
  - Does not report several key Oracle features
  - Does not store collected data in permanent tables
  - Does not separate collection from reporting

# What is Statspack?


Collects More Data
Pre-Calculates Performance Ratios
Stores Performance Statistics
References Historical Data
*Statspack Features*

# StatsPack

- Set of SQL, PL/SQL, and SQL*Plus scripts for collection, automation, storage, and viewing of performance data
- Succeeds BSTAT/ESTAT
- Available since version 8.1.6 (works with 8.x, but is not supported)
- Features
  - Identifies top wait events
  - Collects more information than BSTAT/ESTAT
  - Pre-calculates statistical ratios
  - Uses permanent tables
  - Separates reporting and collection events
  - Supports automatic snapshots
  - Counts commits and rollbacks as finished transactions

# Comparison of BSTAT/ESTAT and StatsPack

| Feature | BSTAT/ESTAT | StatsPack |
|---|---|---|
| Instance summary page | NO | YES |
| Normalization of instance statistics by time and number of transactions | NO | YES |
| Wait events | YES | YES |
| High-resource SQL | NO | YES |
| Instance-activity statistics | YES | YES |
| Tablespace and file I/O statistics | YES | YES |
| Buffer wait breakdown by type | YES | YES |
| Enqueue statistics | NO | YES |
| Rollback segment activity and storage data | YES | YES |
| Latch activity | YES | YES |
| Latch sleep breakdown | NO | YES |

# Comparison of BSTAT/ESTAT and StatsPack

| Feature | BSTAT/ESTAT | StatsPack |
|---|---|---|
| Latch children | NO | YES |
| Buffer pool statistics | NO | YES |
| Dictionary cache activity | YES | YES |
| Library cache activity | YES | YES |
| SGA memory summary | NO | YES |
| SGA memory breakdown | NO | YES |
| Non-default init.ora parameters | YES | YES |
| Configurable output file | NO | YES |
| Ability to move performance data | NO | YES |
| Configurable amount of data collected | NO | YES |
| Ability to run in multiple instances of OPS/RAC | NO | YES |

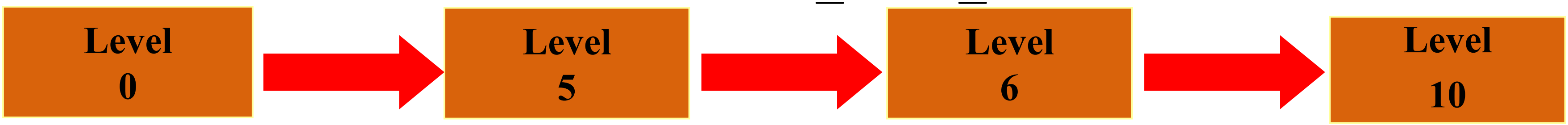
# Snapshot IDs

- Each snapshot is given a sequentially generated snapshot ID
- Snapshot ID, instance number, and database identifier form unique key for the `stat$` tables
- To generate snapshots, execute these SQLPlus commands
  - `connect perfstat/<password>`
  - `execute statspack.snap`
- Do NOT
  - Shut down the instance between snapshots
  - Change the `timed_statistics` settings

# Levels and Thresholds

- Define how much data is collected
- Stored in the `stats$statspack_parameter` table
- To change default settings, execute the `statspack.snap` package with appropriate parameters

```
execute statspack.snap(i_snap_level=>0);
```



Level 0

Level 5

Level 6

Level 10

Level 0:

- Wait statistics
- System events
- System statistics
- Rollback segment data
- Row cache
- SGA
- Background events
- Session events
- Lock, pool, and latch statistics

Level 5:

Level 0 statistics, plus:

- High-resource SQL

Level 6:

New with Oracle9*i*; includes all previous levels, plus

- SQL plans
- Plan usage for high resource SQL

Level 10:

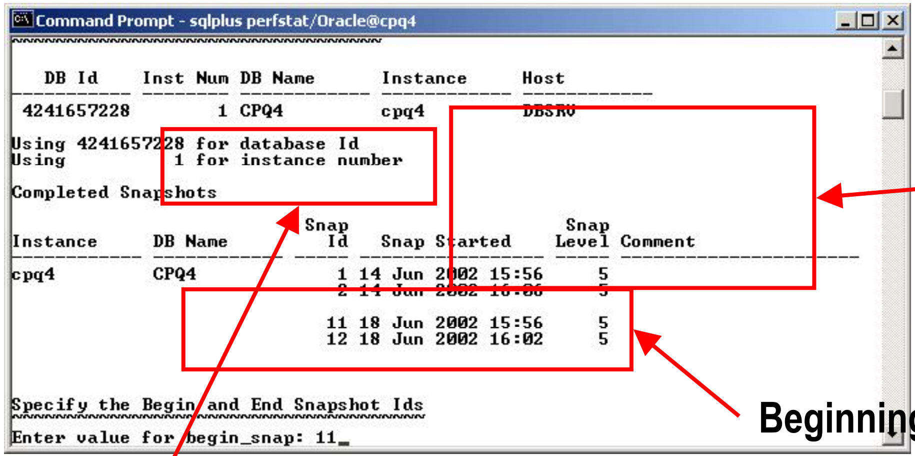
Statistics for all previous levels, plus:

- Child latch data

# Running Reports

- Two reports are available
  - SPREPORT.SQL
    - General instance health and instance performance statistics
    - Connect as perfstat/perfstat
    - Execute `@<ORACLE_HOME>\rdbms\admin\spreport.sql` in SQLPlus
    - Enter beginning and ending snapshot
    - Enter output file name
  - SPREPSQL.SQL
    - Specific SQL statement statistics
    - Run after SPREPORT.SQL
    - Connect as perfstat/perfstat
    - Execute `@<ORACLE_HOME>\rdbms\admin\sprepsql.sql` in SQLPlus
    - Enter beginning and ending snapshot
    - Enter hash value of the specific SQL statement
    - Enter output file name

# Selecting Snapshots

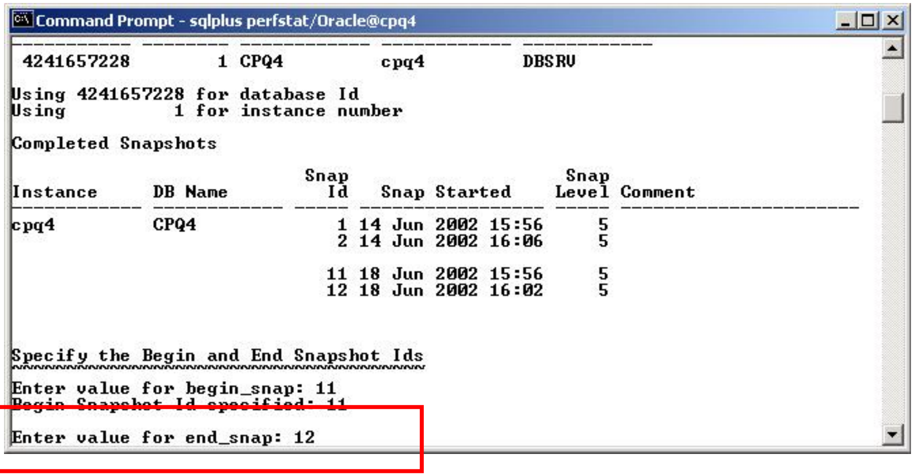


```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   DB Id    Inst Num DB Name      Instance     Host
----------- -------- ------------ ------------ ------------
 4241657228        1 CPQ4         cpq4         DBSRV

Using 4241657228 for database Id
Using          1 for instance number

Completed Snapshots

                                         Snap                   Snap
Instance     DB Name                      Id   Snap Started     Level Comment
------------ ------------ --------------- ----- ----------------- ----- --------------------
cpq4         CPQ4                             1 14 Jun 2002 15:56     5
                                              2 14 Jun 2002 16:06     5

                                             11 18 Jun 2002 15:56     5
                                             12 18 Jun 2002 16:02     5

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 11_
```

Available snapshots

Beginning snapshot

Instance and database information



```
----------- -------- ------------ ------------ ------------
 4241657228        1 CPQ4         cpq4         DBSRV

Using 4241657228 for database Id
Using          1 for instance number

Completed Snapshots

                                         Snap                   Snap
Instance     DB Name                      Id   Snap Started     Level Comment
------------ ------------ --------------- ----- ----------------- ----- --------------------
cpq4         CPQ4                             1 14 Jun 2002 15:56     5
                                              2 14 Jun 2002 16:06     5

                                             11 18 Jun 2002 15:56     5
                                             12 18 Jun 2002 16:02     5

Specify the Begin and End Snapshot Ids
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Enter value for begin_snap: 11
Begin Snapshot Id specified: 11

Enter value for end_snap: 12
```

Ending snapshot

# Performance or Perception ?

- End Users – Extended Load window unacceptable
- DBA – No control over aggregation queries
- System Administrators – Swapping

# Oracle Observations - Wait Events in Oracle

```
Top 5 Wait Events
~~~~~~~~~~~~~~~~~                         Wait     % Total
Event                           Waits  Time (cs)   Wt Time
------------------------- ------------ ------------ --------
PX Deq: Execution Msg         210,637   27,095,090   45.13
PX Deq Credit: send blkd      254,306   20,185,656   33.62
PX Deq: Table Q Normal        284,330    8,989,133   14.97
PX Deq: Execute Reply          53,477    2,160,686    3.60
direct path read              261,773      531,092     .88
                                    ----------------
Wait Events for DB: BIWP  Instance: BIWP  Snaps: 7950 -7957
-> cs - centisecond -  100th of a second
-> ms - millisecond - 1000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

Wait Events consistent with earlier statspack

# Response Time Analysis

Response Time = Service Time + Wait Time

- Queuing Theory
- Big Picture -> Service or Waits ?

```
  Statistic                              Total   per Second    per Trans
--------------------------------- ---------------- ------------ ------------
CPU used by this session              1,464,542        102.5        133.7
```

Service Time = 1,464,542 Centiseconds

# Response Time Analysis

## Top 5 Wait Events

```
~~~~~~~~~~~~~~~~~~~                                   Wait      % Total
   Event                                   Waits  Time (cs)    Wt Time
-------------------------------------------- ------------ ------------ -------
PX Deq: Execution Msg                      210,637   27,095,090   45.13
PX Deq Credit: send blkd                   254,306   20,185,656   33.62
PX Deq: Table Q Normal                     284,330    8,989,133   14.97
 PX Deq: Execute Reply                      53,477    2,160,686    3.60
   direct path read                        261,773      531,092     .88
```

Total Wait Time = 27095090 * 100 /45.13

= 60,037,868 Centi-Seconds

# Response Time Analysis

Response Time = Service Time + Wait Time

= 1,464,542 + 60,037,868

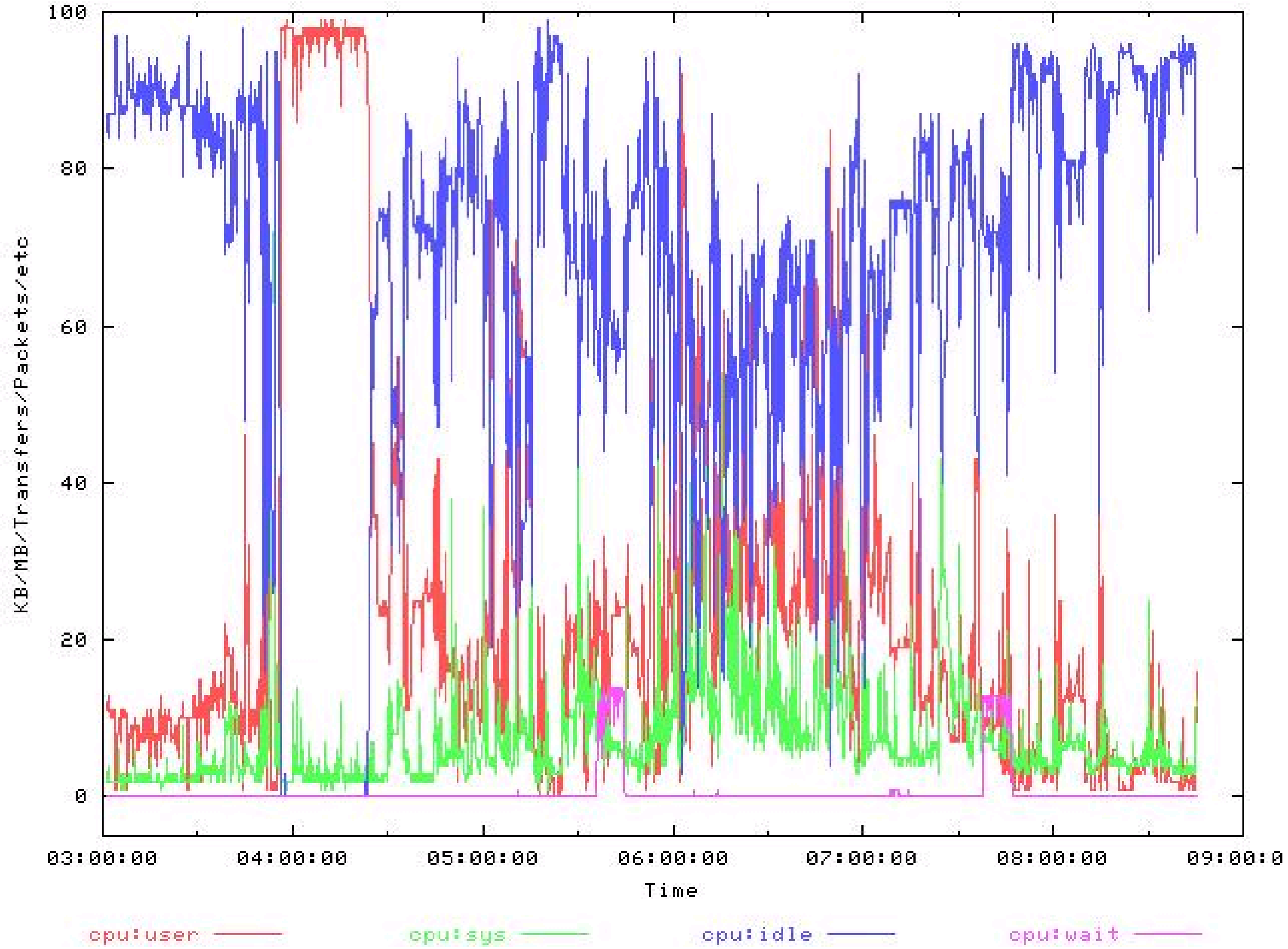
= 61,502,410

% of Response Time

Cpu time = 2.4 %

Wait Time = 97.6 %

# Pre-Tuning Behavior - CPU



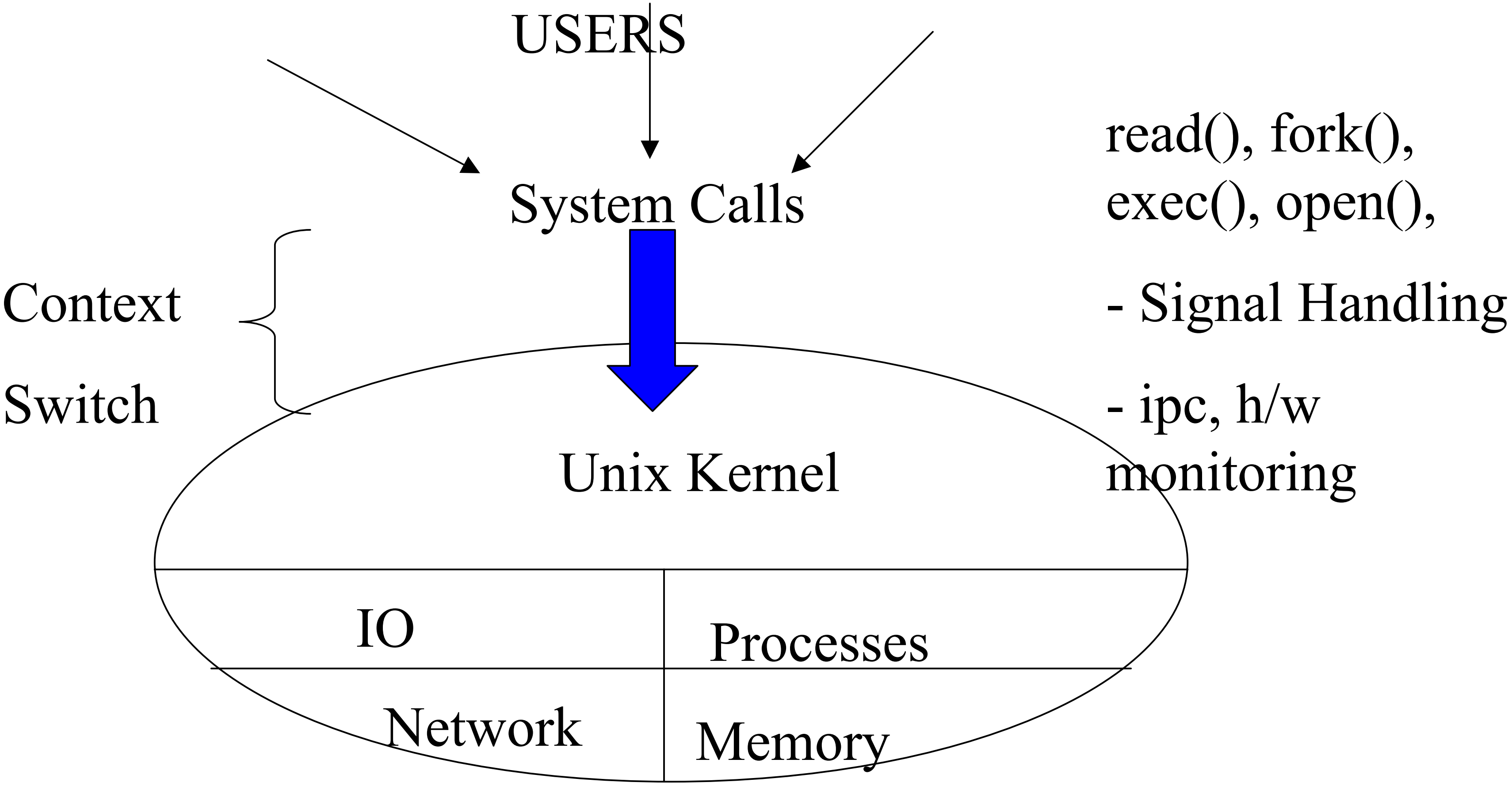
# Observations - CPU

High System Time

- Thrashing
- Insufficient Memory
- Paging / Swapping

# Other Cpu related Observations

- Vmstat (sy) column



USERS

System Calls

read(), fork(), exec(), open(),

- Signal Handling

- ipc, h/w monitoring

Context Switch

Unix Kernel

IO

Processes

Network

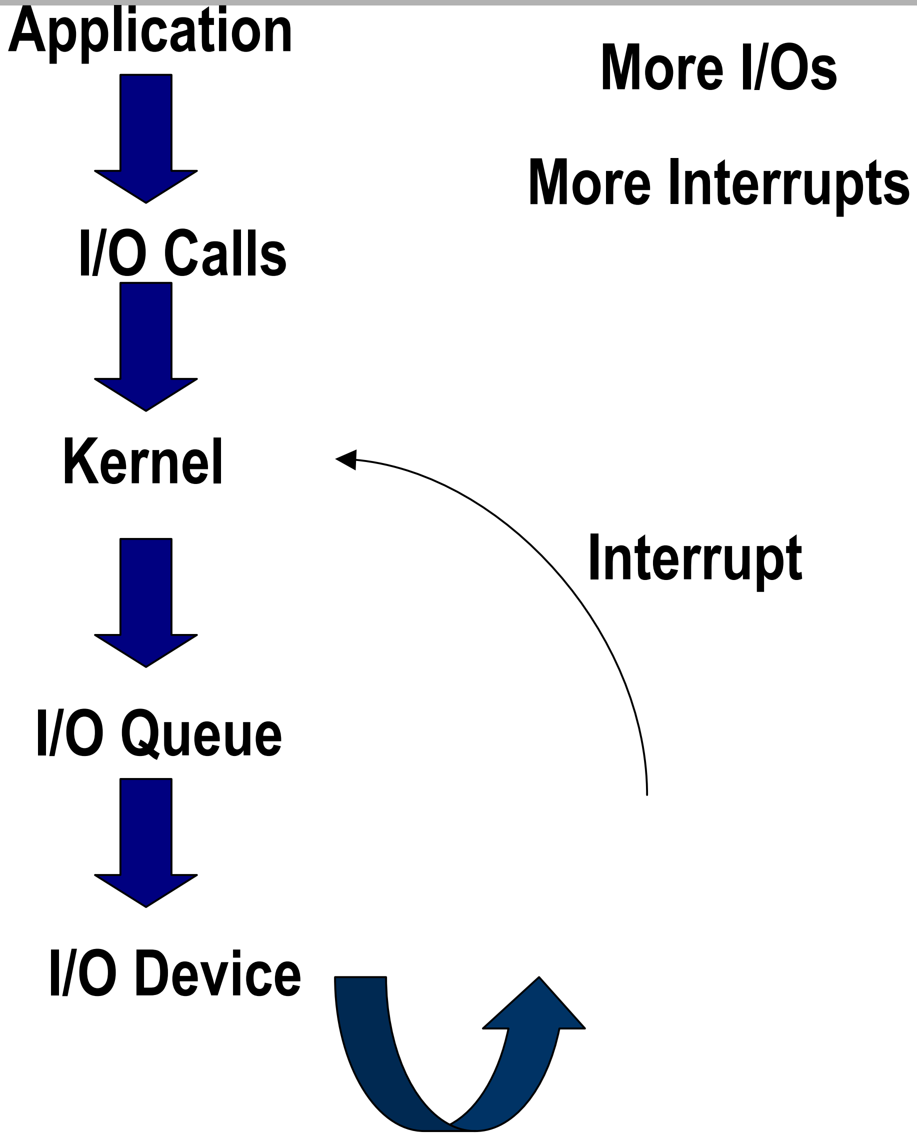
Memory

# Other Cpu related Observations

## System Calls

- Goal is to Minimize the system calls, Context Switches
- Reduce Time spent in Kernel
- Blocking I/O (Synchronous)
- Non-Blocking I/O (Asynchronous)

# Interrupts



Application

I/O Calls

Kernel

I/O Queue

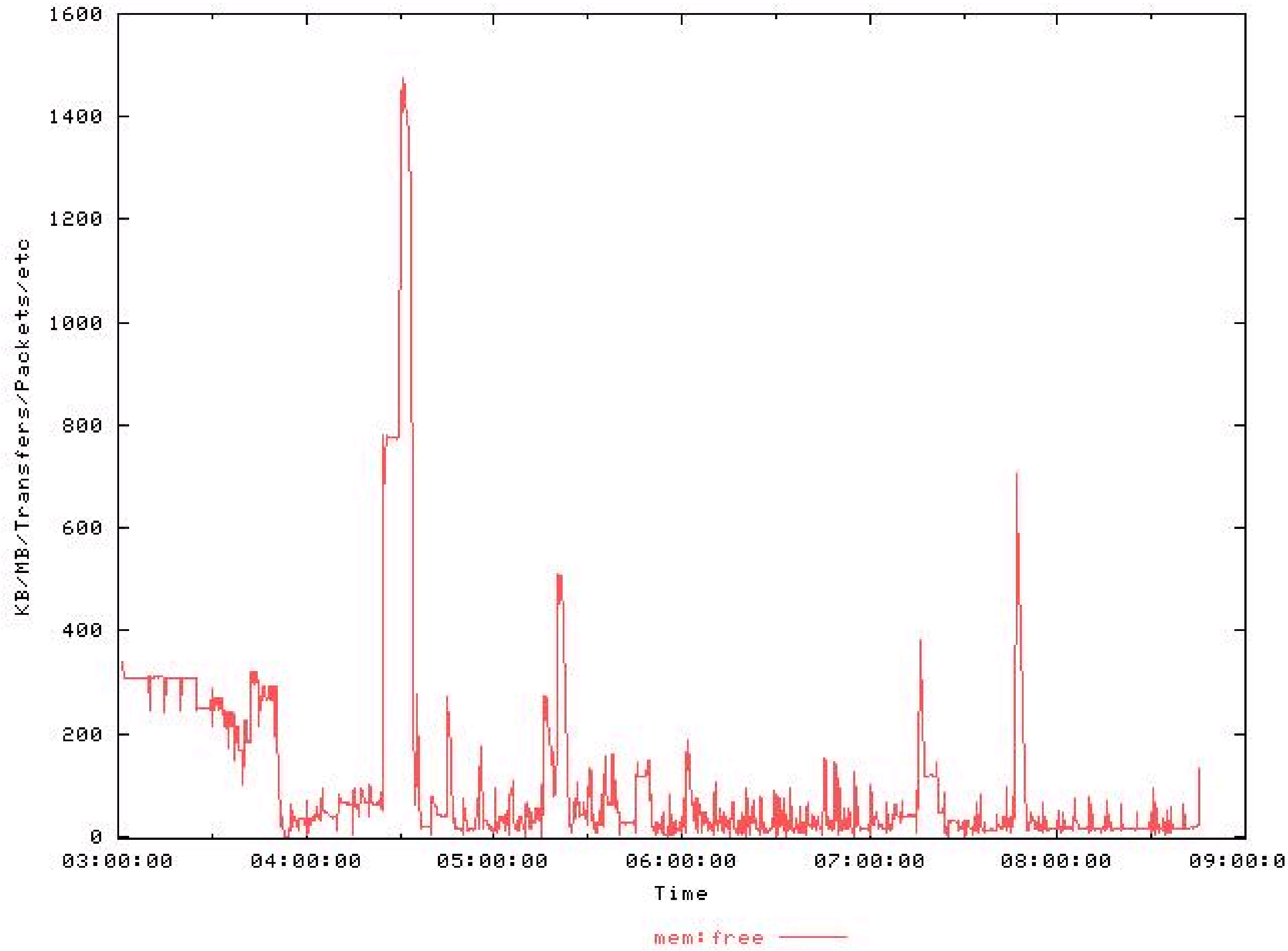
I/O Device

Interrupt

More I/Os

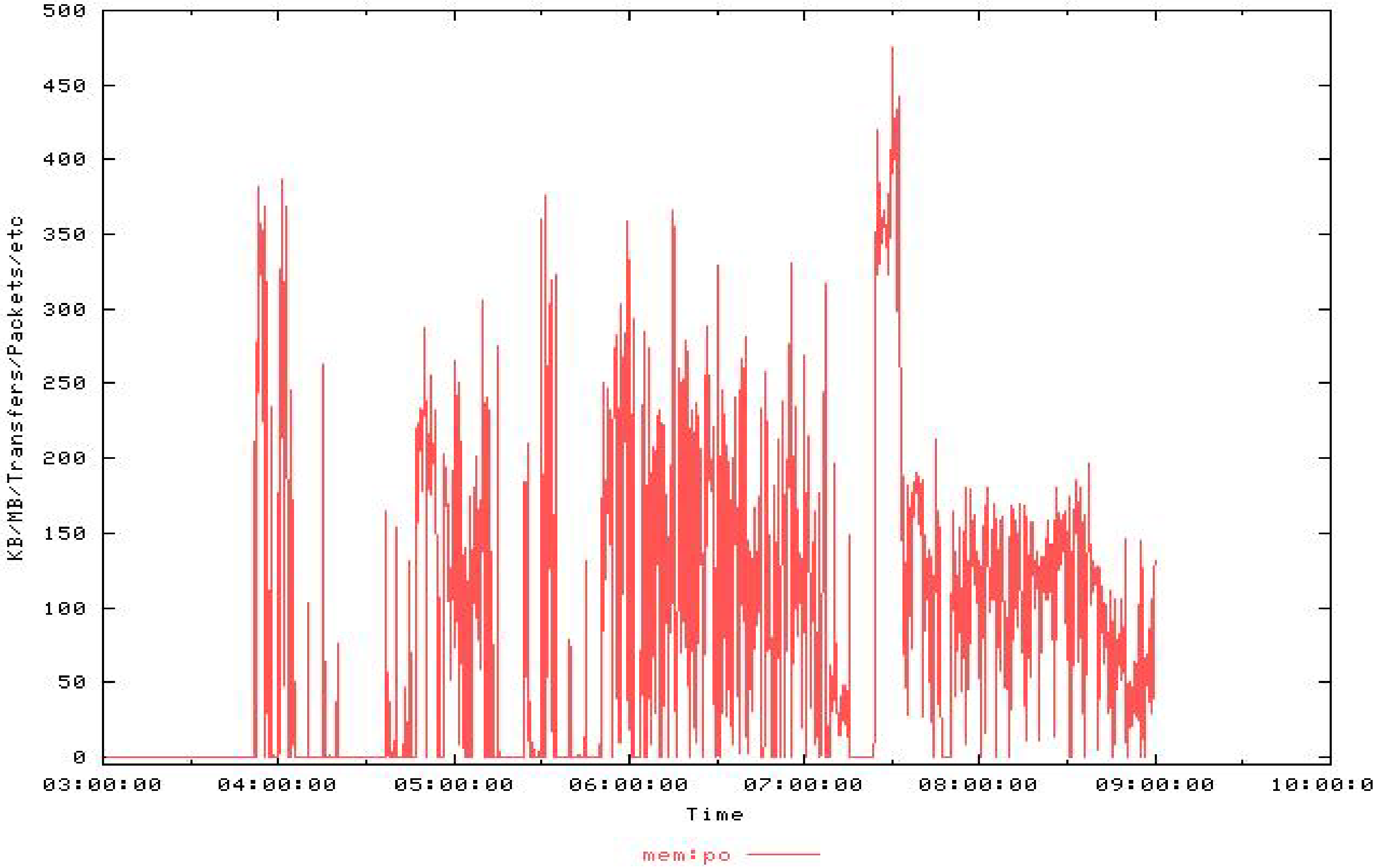
More Interrupts

# Memory – Free

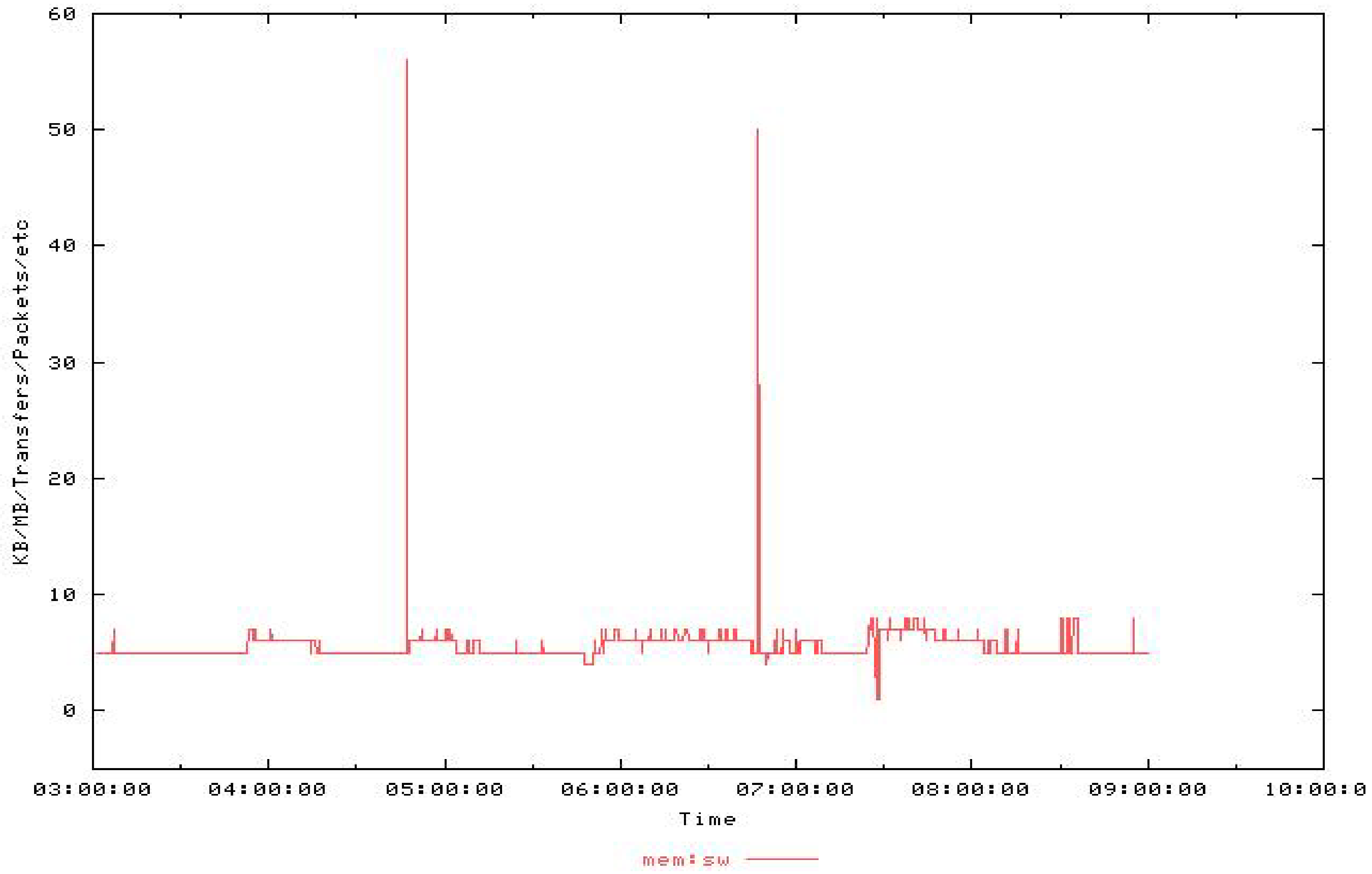


1600
1400
1200
1000
800
600
400
200
0
KB/MB/Transfers/Packets/etc
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:0
Time
mem:free

# Memory – Page Outs/Sec



# Memory – No. of Swapped Processes

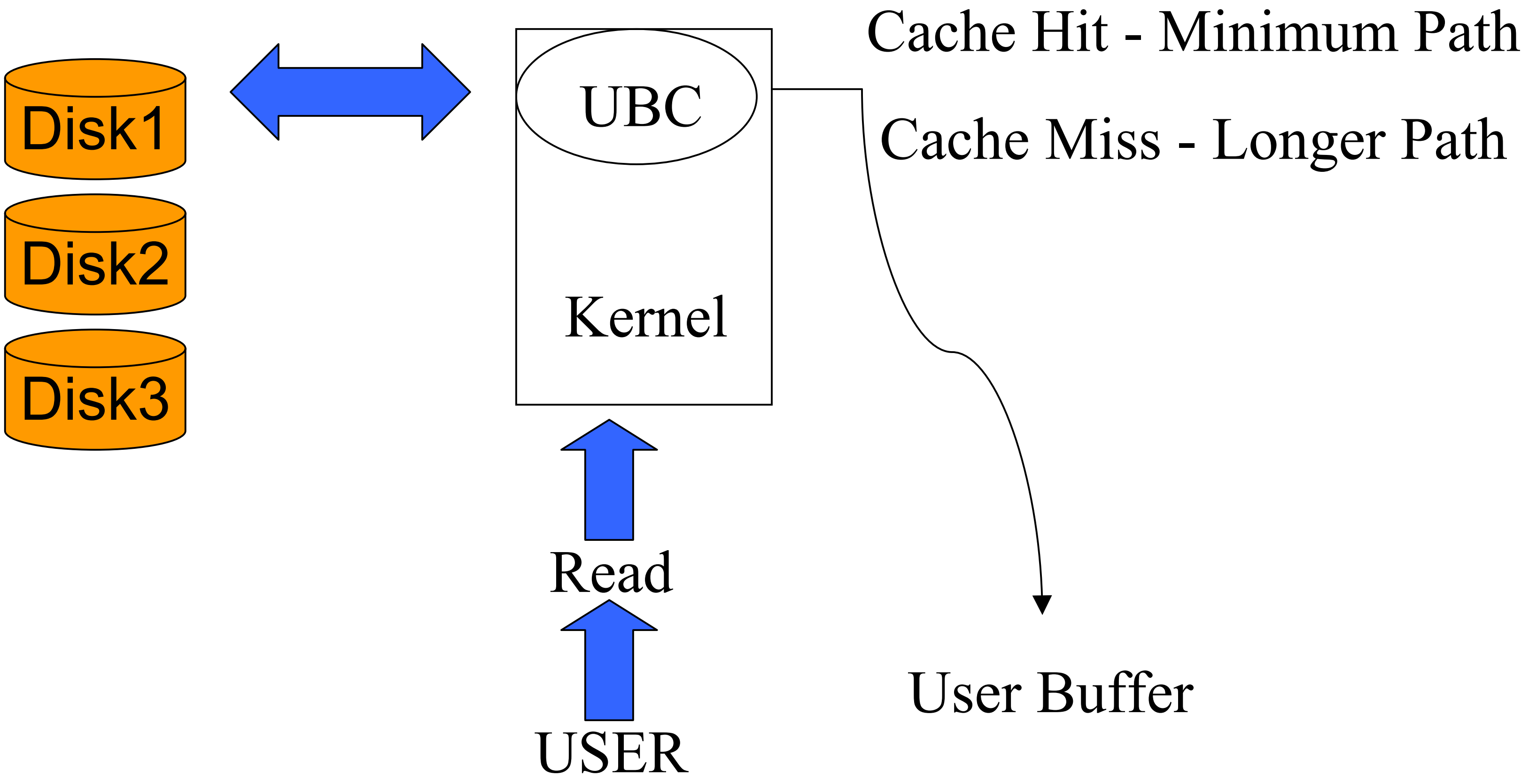


# File and Disk Subsystem

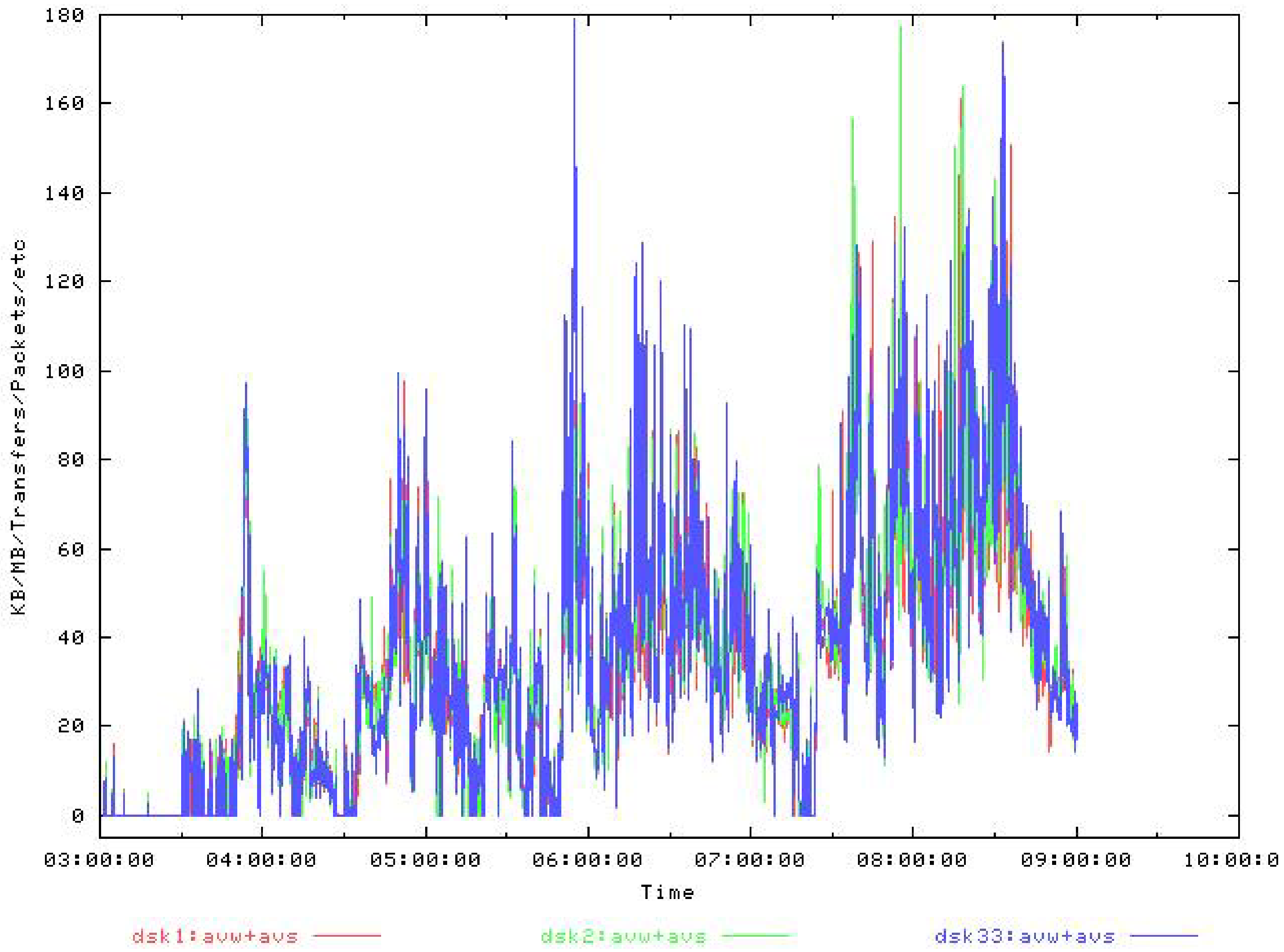
File System

- File System Cache
- Double Buffering (SGA & FS Cache)
- No such issue with raw devices and Direct IO

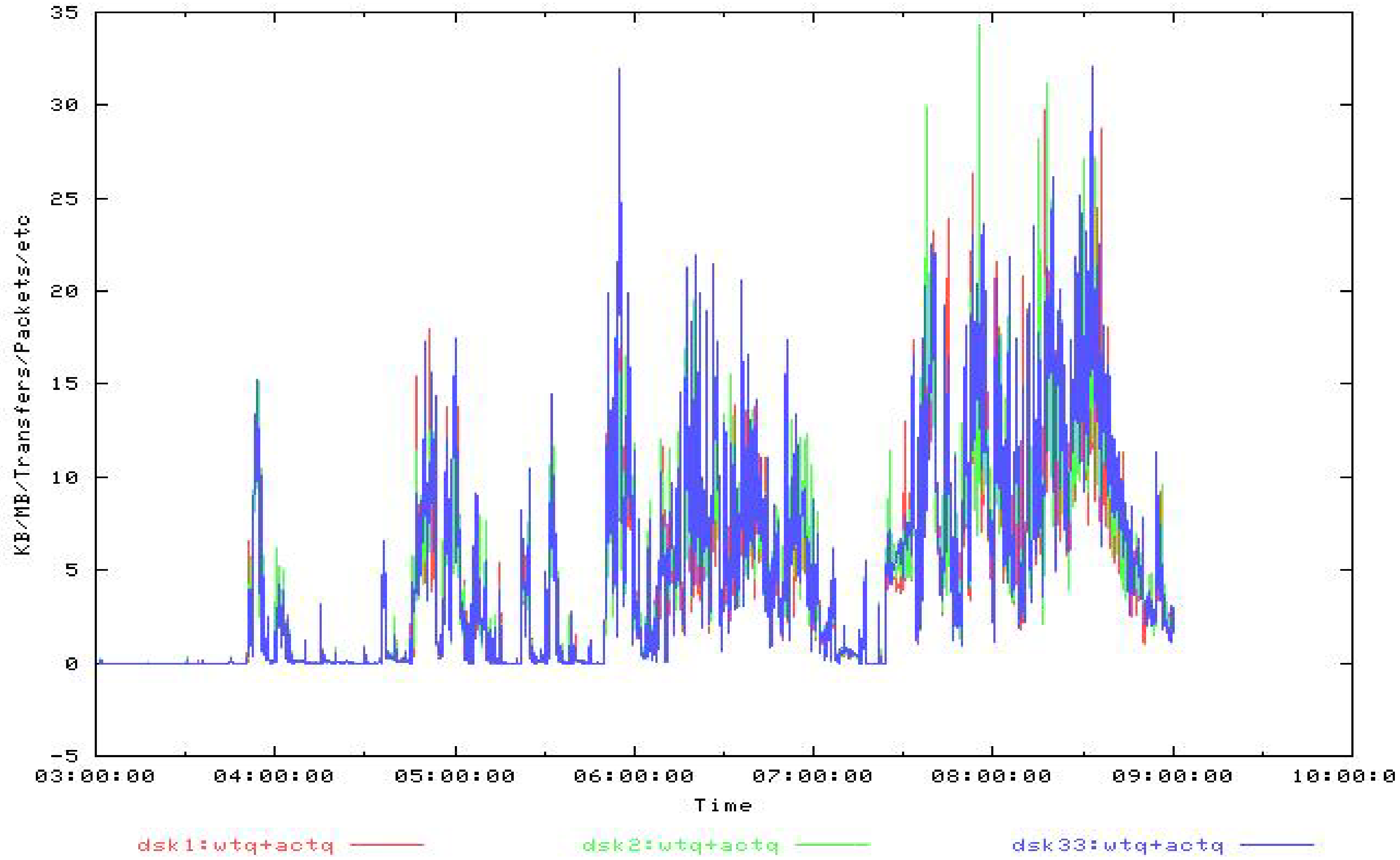
# I/O Operation



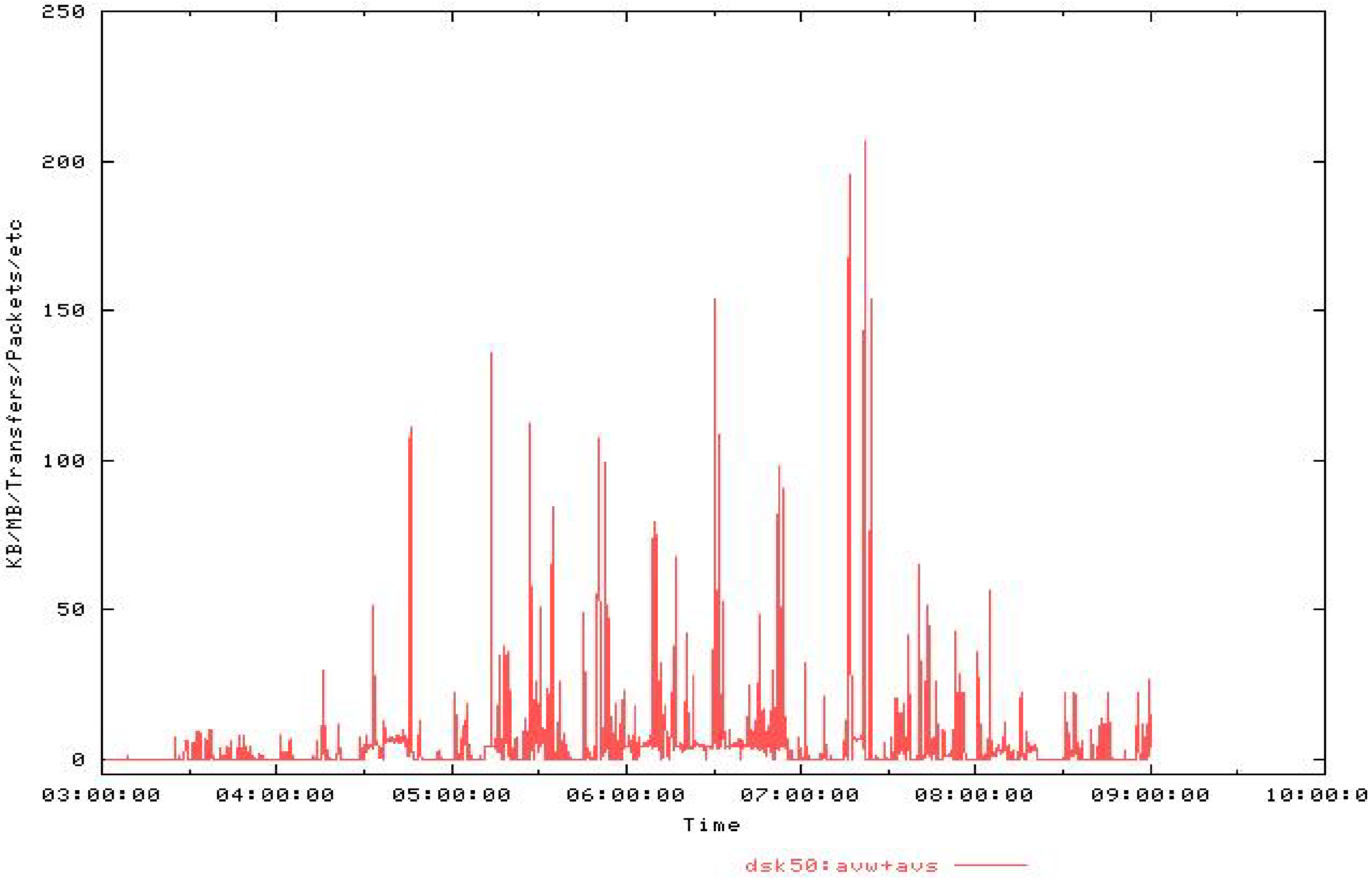
# Swap Disks average service times (dsk1,dsk2,dsk33)



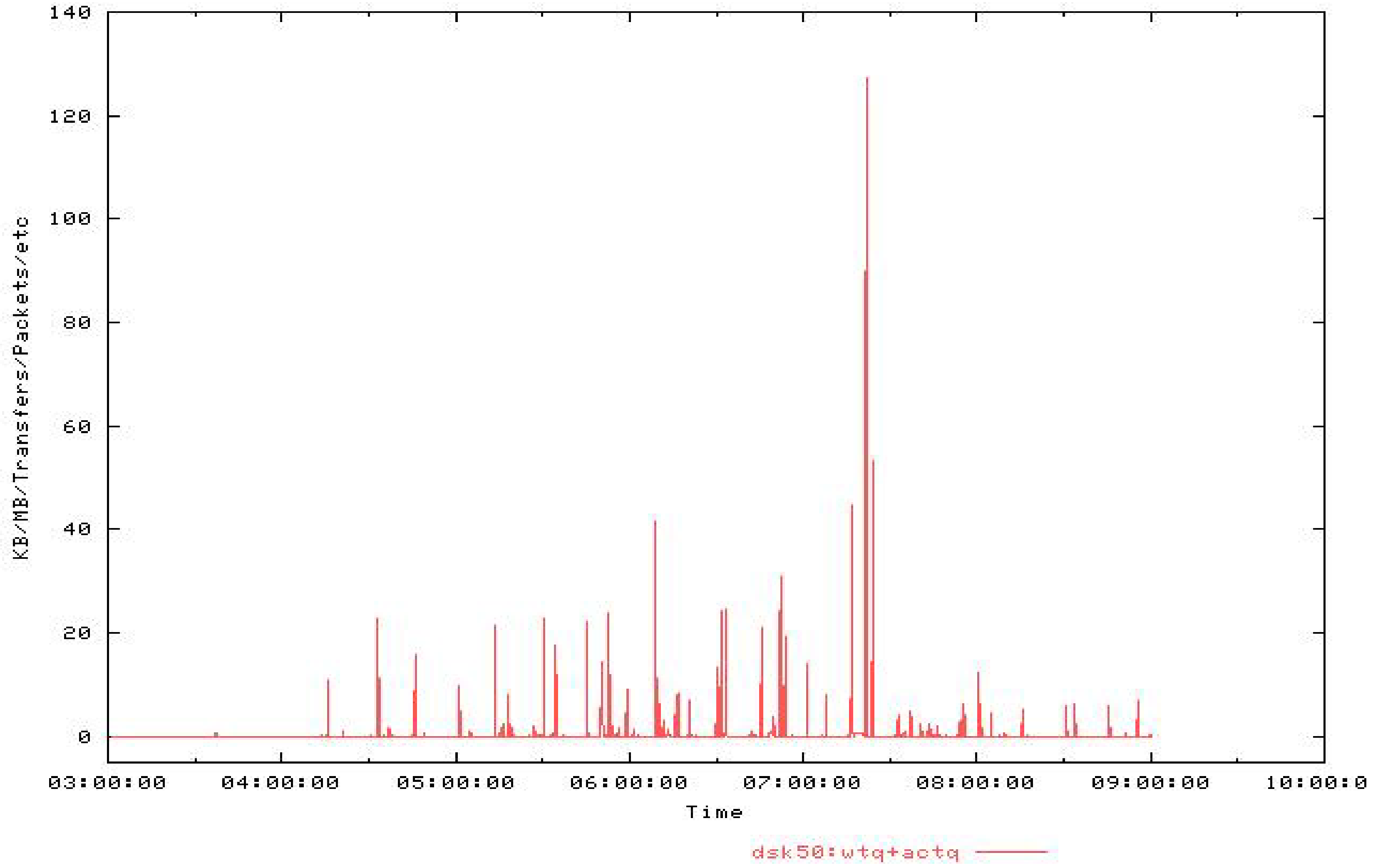
# Same Disks with Total queue lengths



# Remaining Disks with Hotspots (dsk50) - AVS



# dsk50 with total queue lengths



# PX Deq: Execution Msg

- This event appears when a PQ slave has nothing to do, but is not allowed to go idle.
  E.g. Large data set coming out of a parallel ORDER BY.

- The last layer of PX slaves in the query will receive a ranged set of rows and sort them. The QC will then request ALL the rows from the first PX slave, then the second, then the third and so on.

- usually an idle event – but it may be a symptom of excessively large queries choosing an inappropriate execution path. (Bad SQL)

# Hash Partitioning/Cardinality

•Affects Distribution

•The number of partitions should always be a power of two (2, 4, 8, and so on)to obtain the most even data distribution.

•Cardinality is important for choosing a key for hash

partitioning.

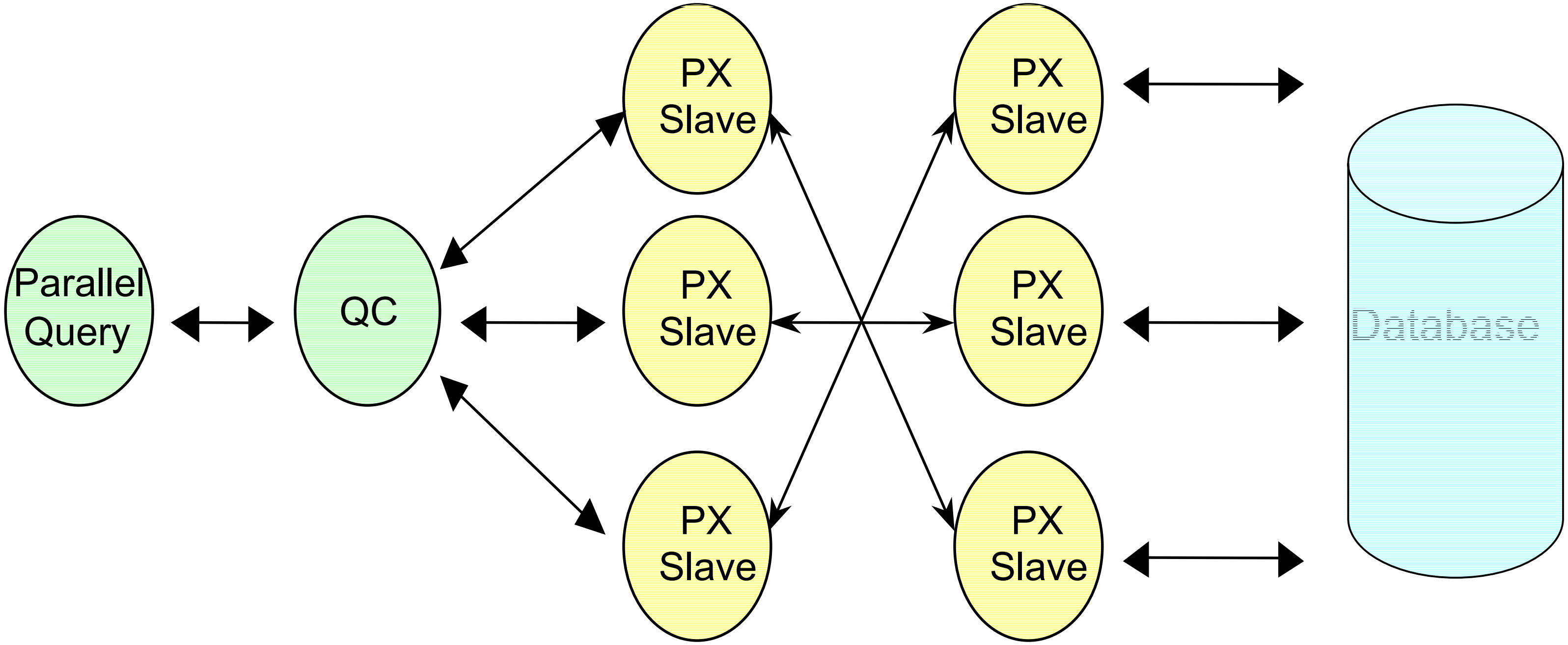
•Oracle recommendation – Do not use partitioning on key columns with low cardinality

•(columns in which the number of distinct values is small compared to the number of rows in the table.)

# Hash Partitioning

- The hash function works best with a LARGE number of values.
- Rows are mapped into partitions based on the hash value of the partitioning key.
- A primary key is a very good hash key because it has a wide range a values.

# PQO


Parallel Query
QC
PX Slave
PX Slave
PX Slave
PX Slave
PX Slave
PX Slave
Database

# Degree of Parallelism

- max_parallel_servers set to 80
- Degree of parallelism set to 16 on key tables.
- parallel_threads_per_cpu set to 4
- parallel_automatic_tuning set to true
- parallel_broadcast_enabled set to true

# Parallel_threads_per_cpu

- This parameter is used to adjust the load on each CPU when PARALLEL_ADAPTIVE_MULTI_USER is enabled.
- The value represents the average number of PX slaves that each CPU can process concurrently..
- If the host system has a few high-powered CPUs rather than many lower performance CPUs, increasing the value may improve throughput.
- Likewise, if the host system has a slower I/O subsystem, increasing the value may improve PX throughput.

# Oracle Observations

- SORT_AREA_SIZE set to 128MB

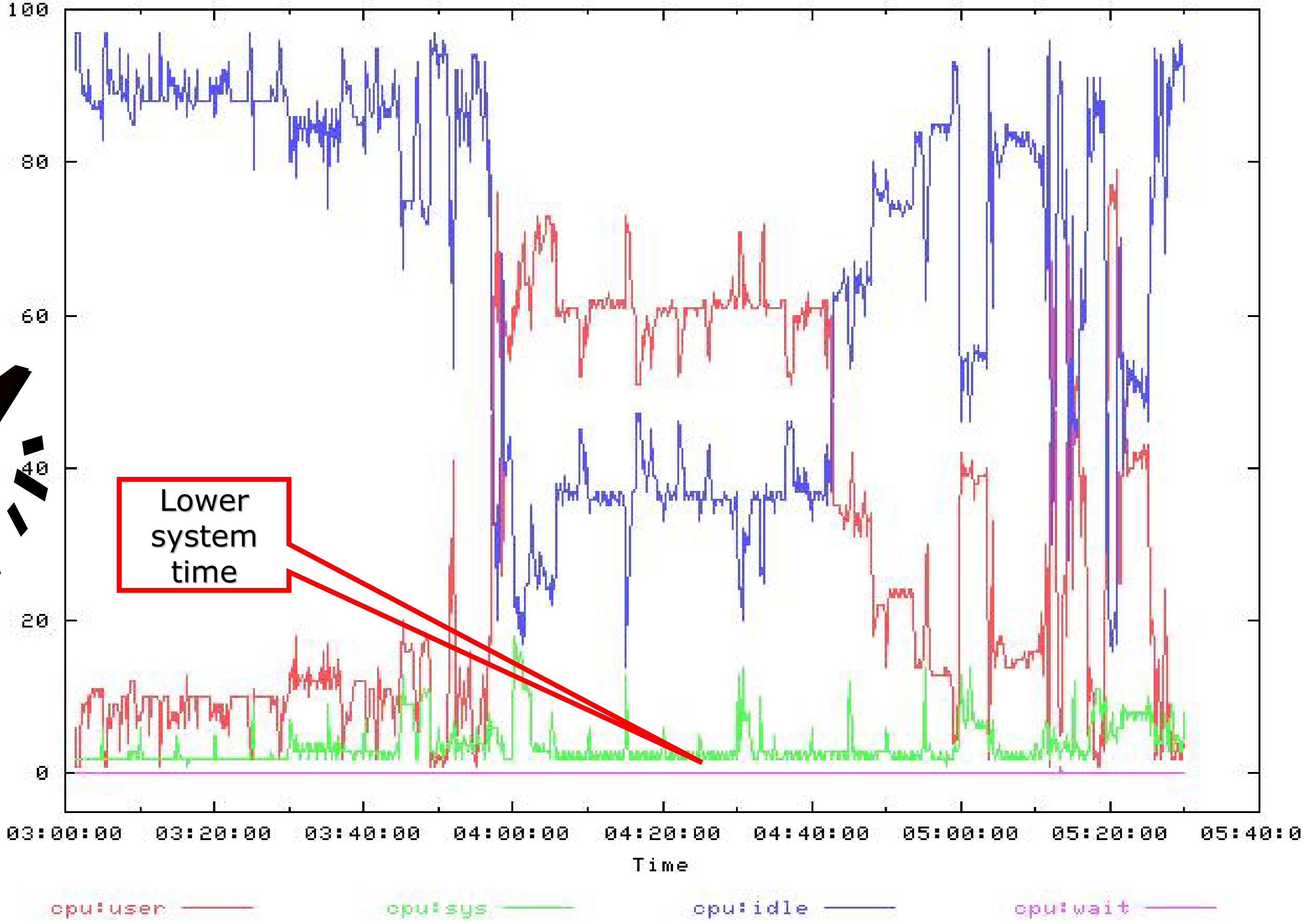
  E.g. 48 parallel query threads could potentially consume up to 6 GB of RAM
- Average no.of Oracle Sessions was around 100
- The average use of PGA per session was around 104 MB
- The wired memory for SGA can be reduced by 3 GB (i.e. from 8 GB to 5 GB) as there was around 15% free SGA (Current setting at 4.5 GB )
- Event 10046 trace pointed to excessive parallel query waits

# Near Term Recommendations

- SORT_AREA_SIZE set to 90 MB (from 128 MB)
- Reduced the degree of parallelism to 4 (from 16)
- The wired memory for SGA was reduced by 3 GB

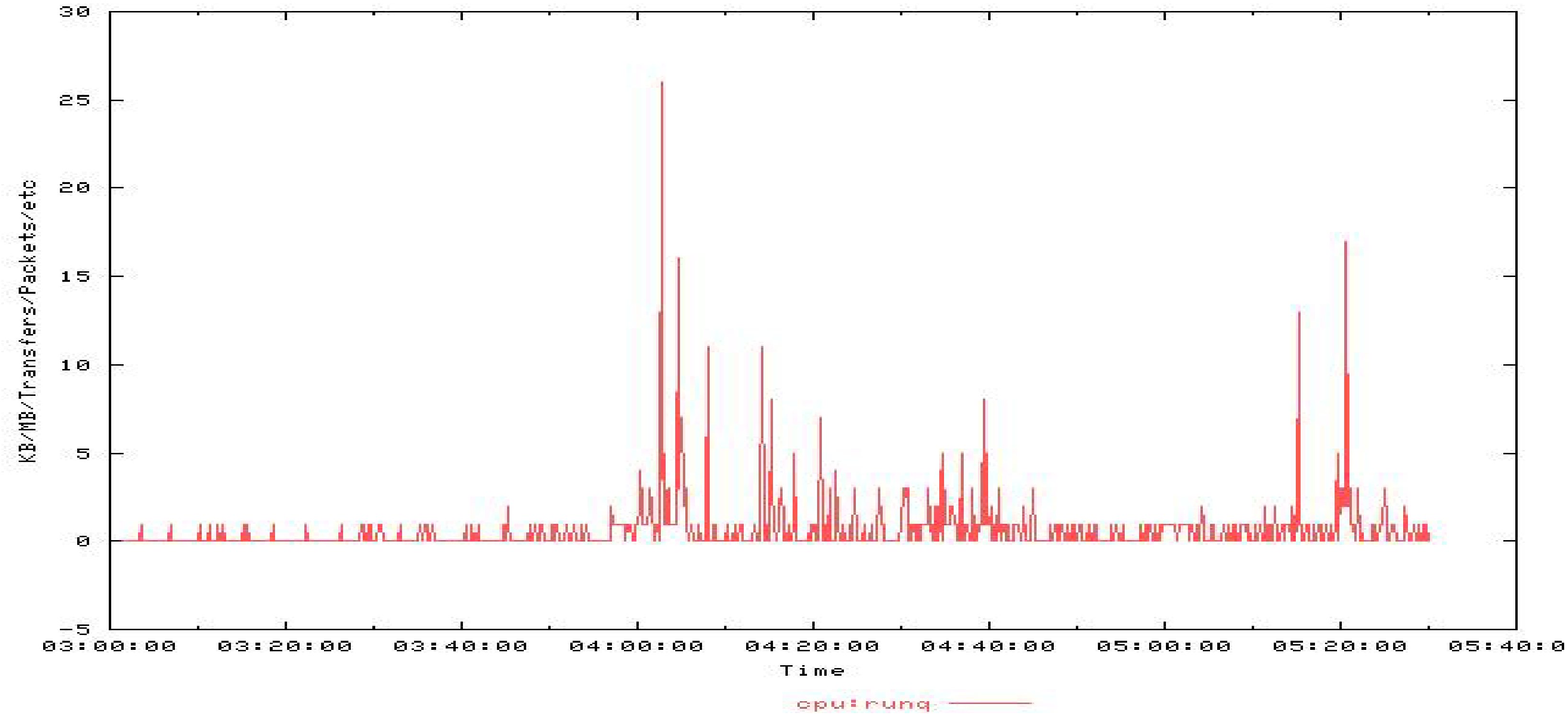
# Post – Tuning behavior (CPU)

- The load process finished at 5:30am !

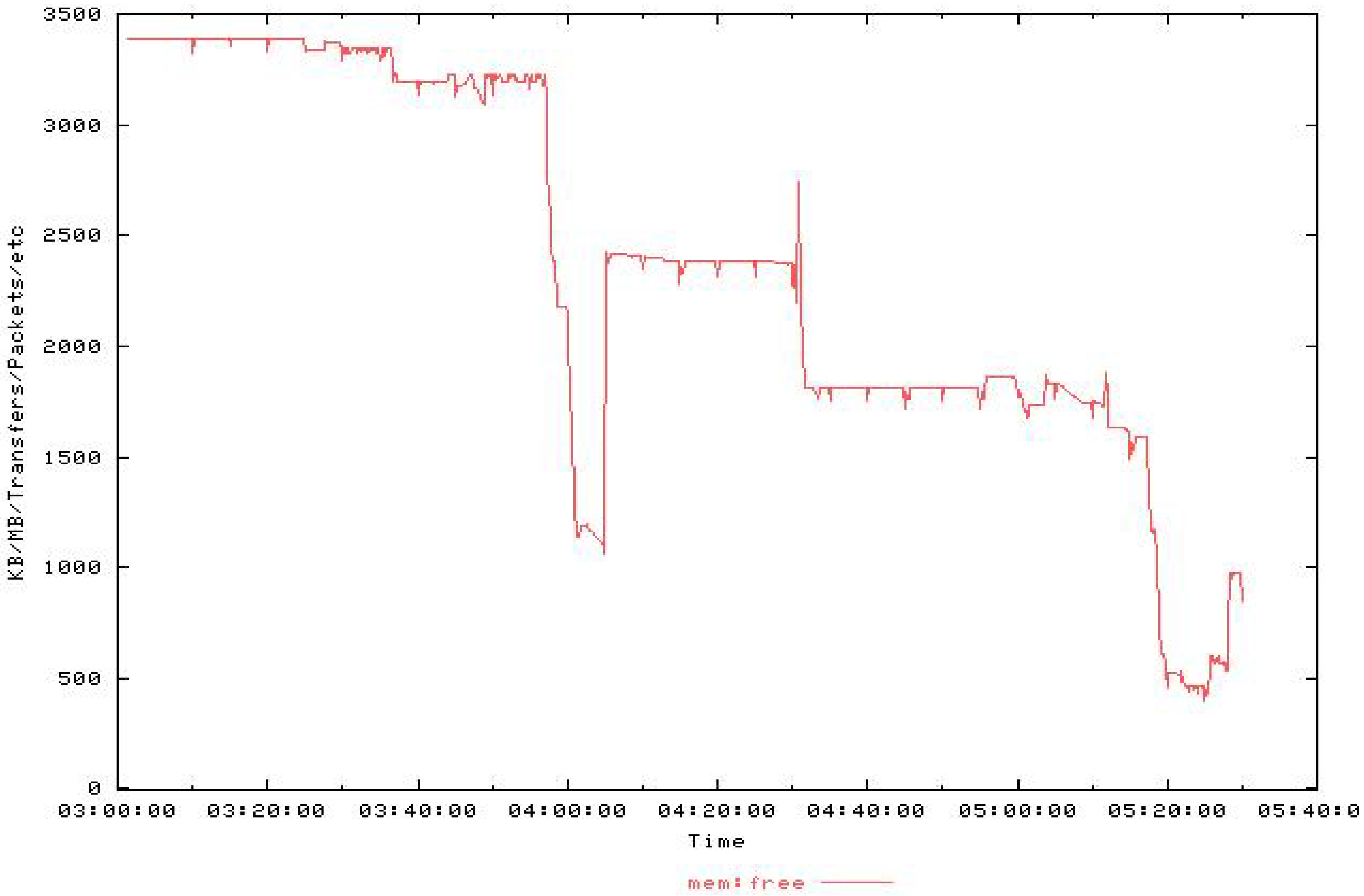


Lower system time

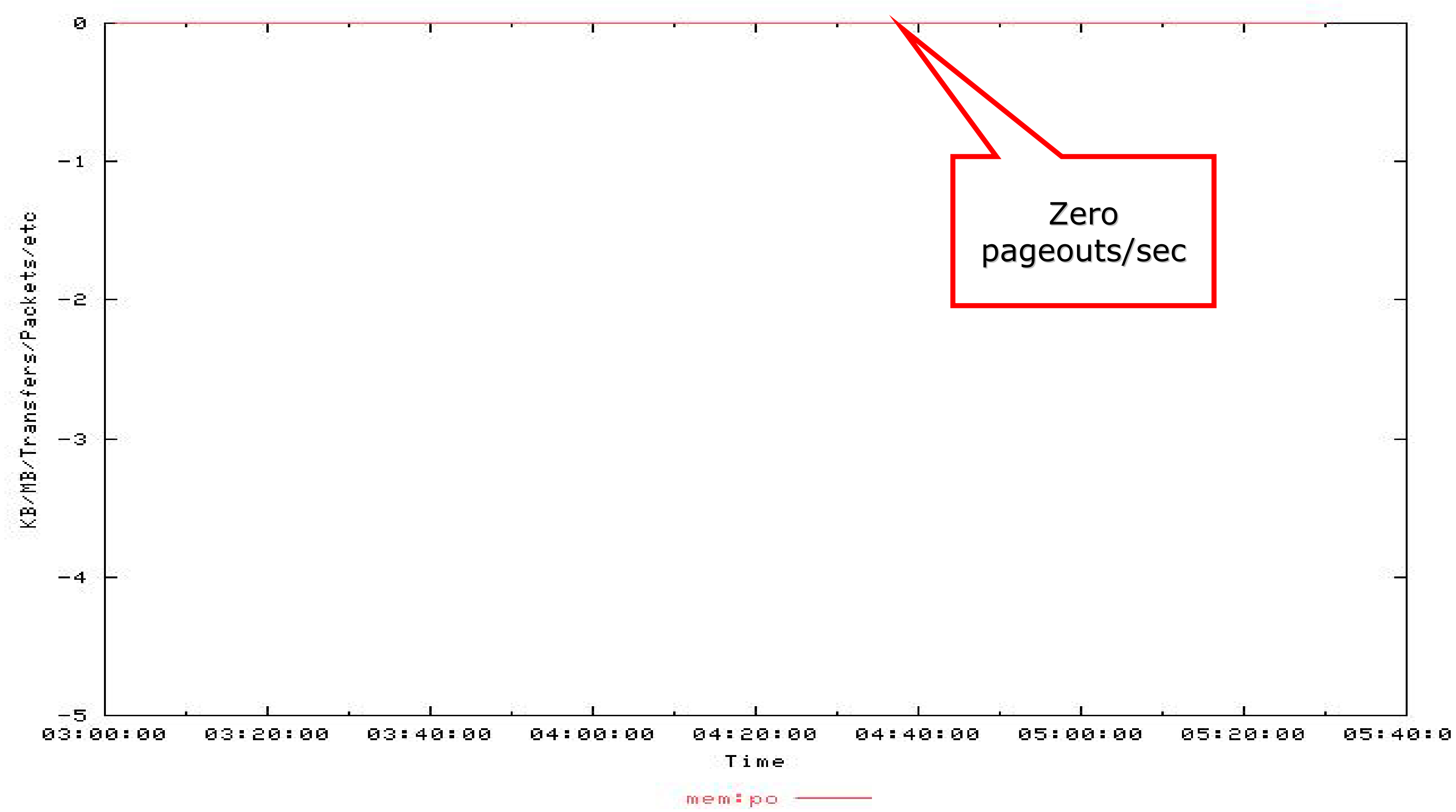
# CPU Observations – Run Queue



# Memory (Free)

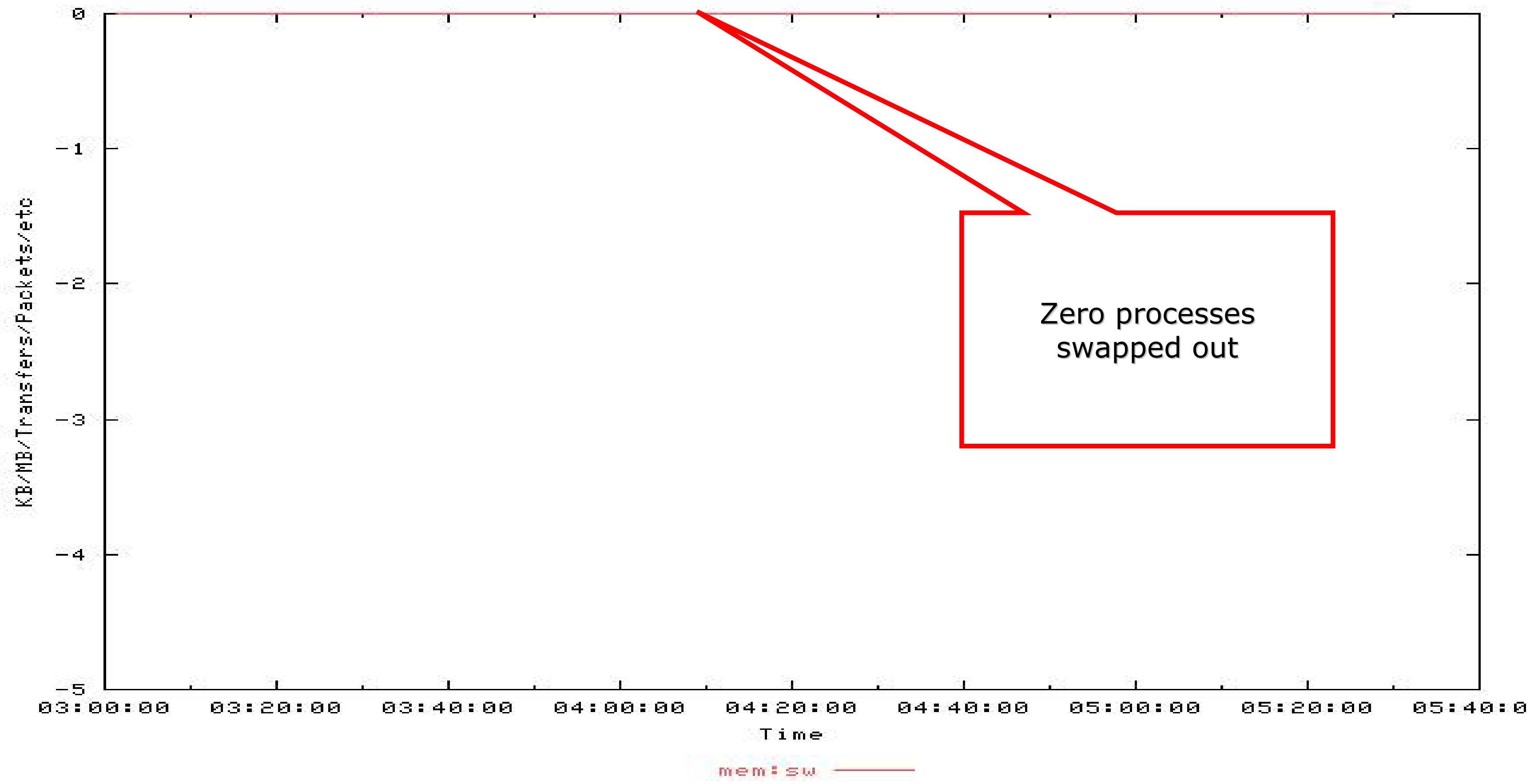

3500
3000
2500
2000
1500
1000
500
0
KB/MB/Transfers/Packets/etc
03:00:00
03:20:00
03:40:00
04:00:00
04:20:00
04:40:00
05:00:00
05:20:00
05:40:0
Time
mem:free

# Memory – Pageouts/sec



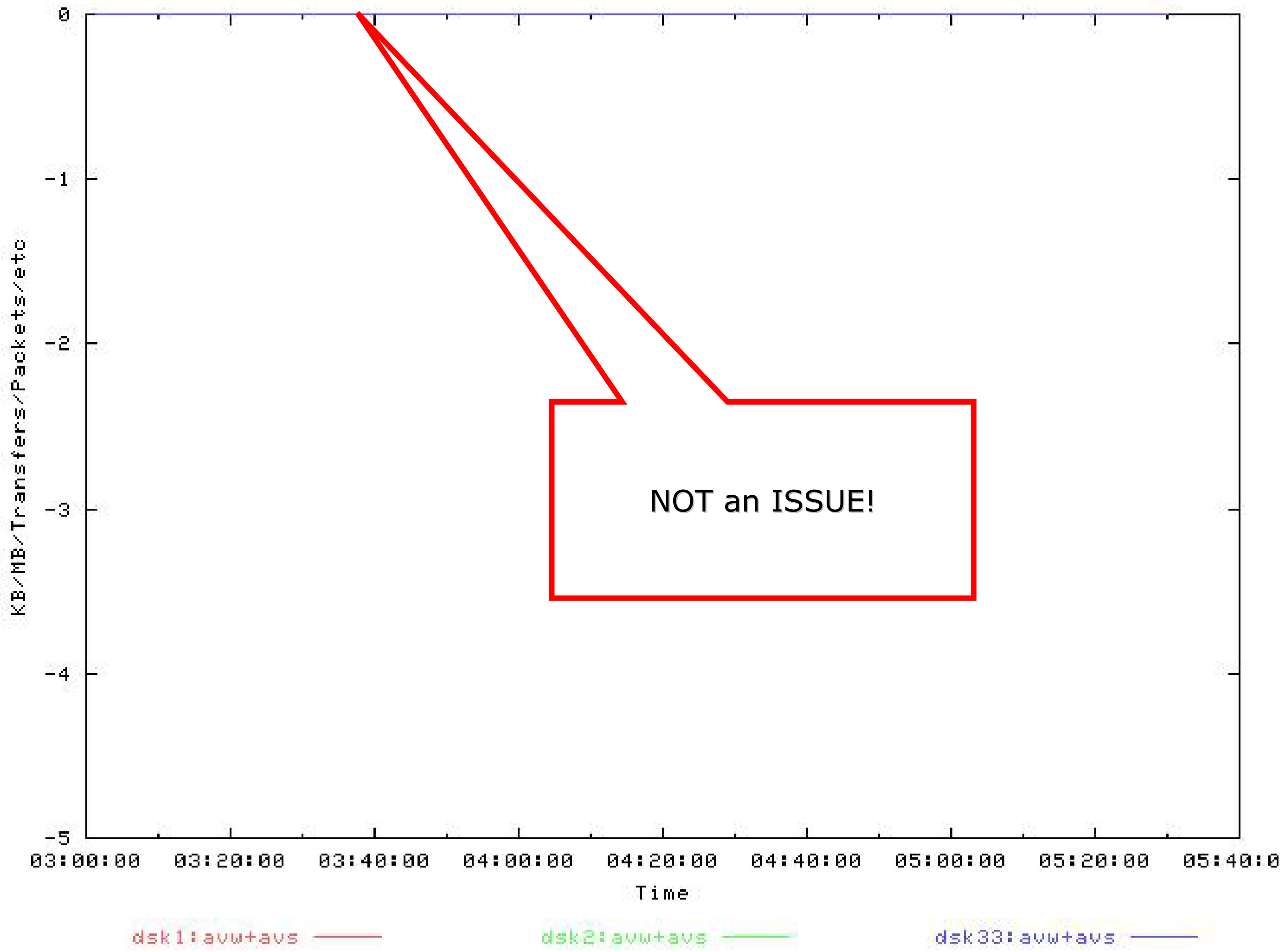
Zero pageouts/sec

# Memory – No. of Swapped out processes



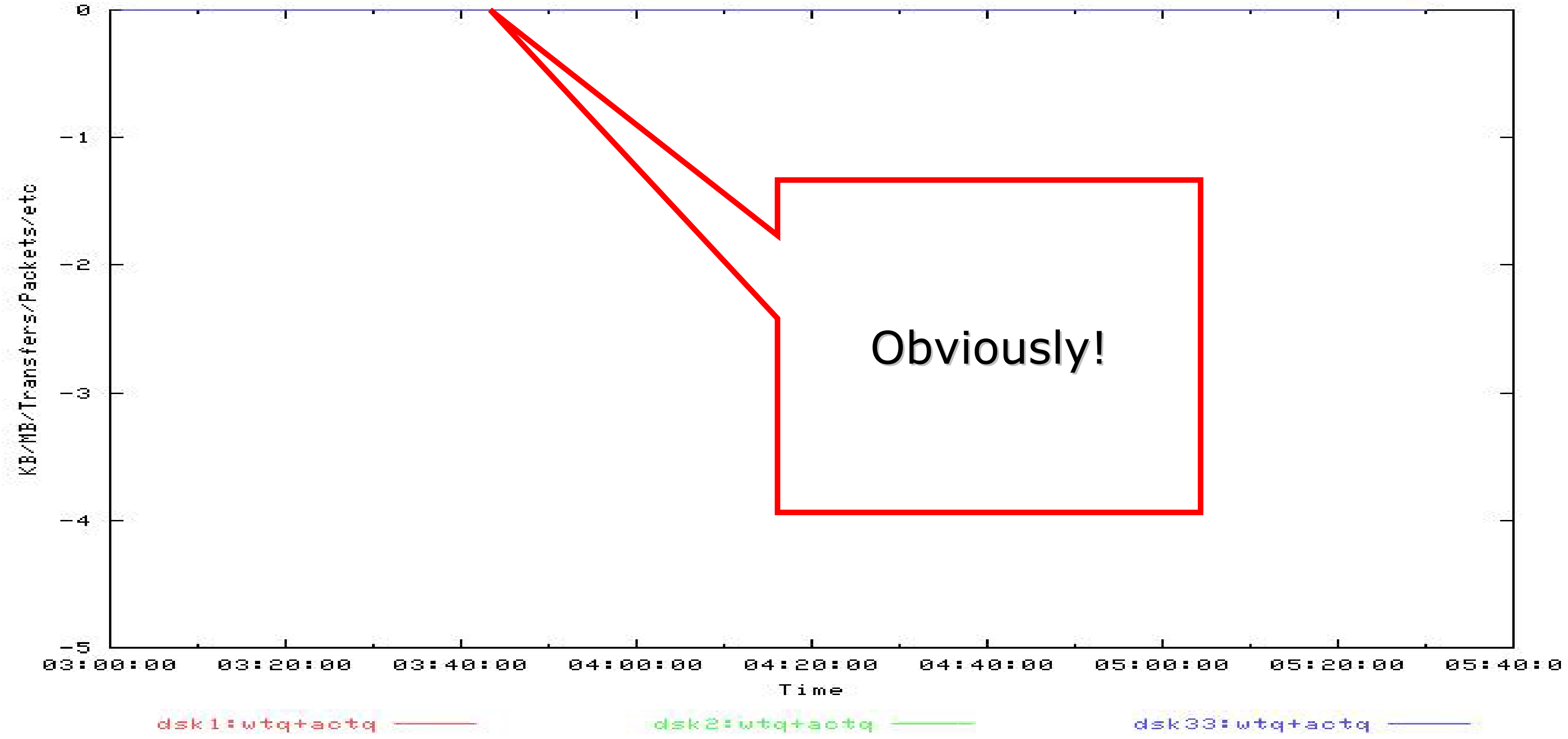
Zero processes swapped out

# Disk – Response times on swap disks (dsk1, 2, 33)



NOT an ISSUE!

# Swap disks queue lengths



Obviously!

# Oracle Observations - Wait Events in Oracle

```
 Top 5 Wait Events
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~                      Wait     % Total
Event                                         Waits  Time (cs)   Wt Time
-------------------------------------------- ------------ ------------ -------
PX Deq: Execution Msg                           362,040    6,246,486   46.72
PX Deq: Table Q Normal                          419,894    3,648,741   27.29
PX Deq Credit: send blkd                        121,110      934,631    6.99
db file sequential read                       1,189,706      644,219    4.82
direct path read                                348,736      513,400    3.84
          -------------------------------------------------------------
□Wait Events for DB: BIWP  Instance: BIWP  Snaps: 7977 -7987
-> cs - centisecond -  100th of a second
-> ms - millisecond - 1000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

Wait Events consistent with earlier statspack

# Response Time Analysis

Response Time = Service Time + Wait Time

```
  Statistic                              Total   per Second    per Trans
--------------------------------- ---------------- ------------ ------------
CPU used by this session                 1,684,418        116.9        182.2
```

Service Time = 1,684,418 Centi-seconds

# Response Time Analysis

```
 Top 5 Wait Events
~~~~~~~~~~~~~~~~~~                                    Wait     % Total
Event                                     Waits  Time (cs)   Wt Time
-------------------------------------------- ------------ ------------ -------
PX Deq: Execution Msg                      362,040    6,246,486   46.72
PX Deq: Table Q Normal                     419,894    3,648,741   27.29
PX Deq Credit: send blkd                   121,110      934,631    6.99
db file sequential read                  1,189,706      644,219    4.82
direct path read                           348,736      513,400    3.84
```

Total Wait Time = 6246486 * 100 /46.72

= 13,370,047 Centi-Seconds

# Response Time Analysis

Response Time = Service Time + Wait Time

= 1,684,418 + 13,370,047

= 15,054,465

% of Response Time

Cpu time = 11.18 %

Wait Time = 88.82 %

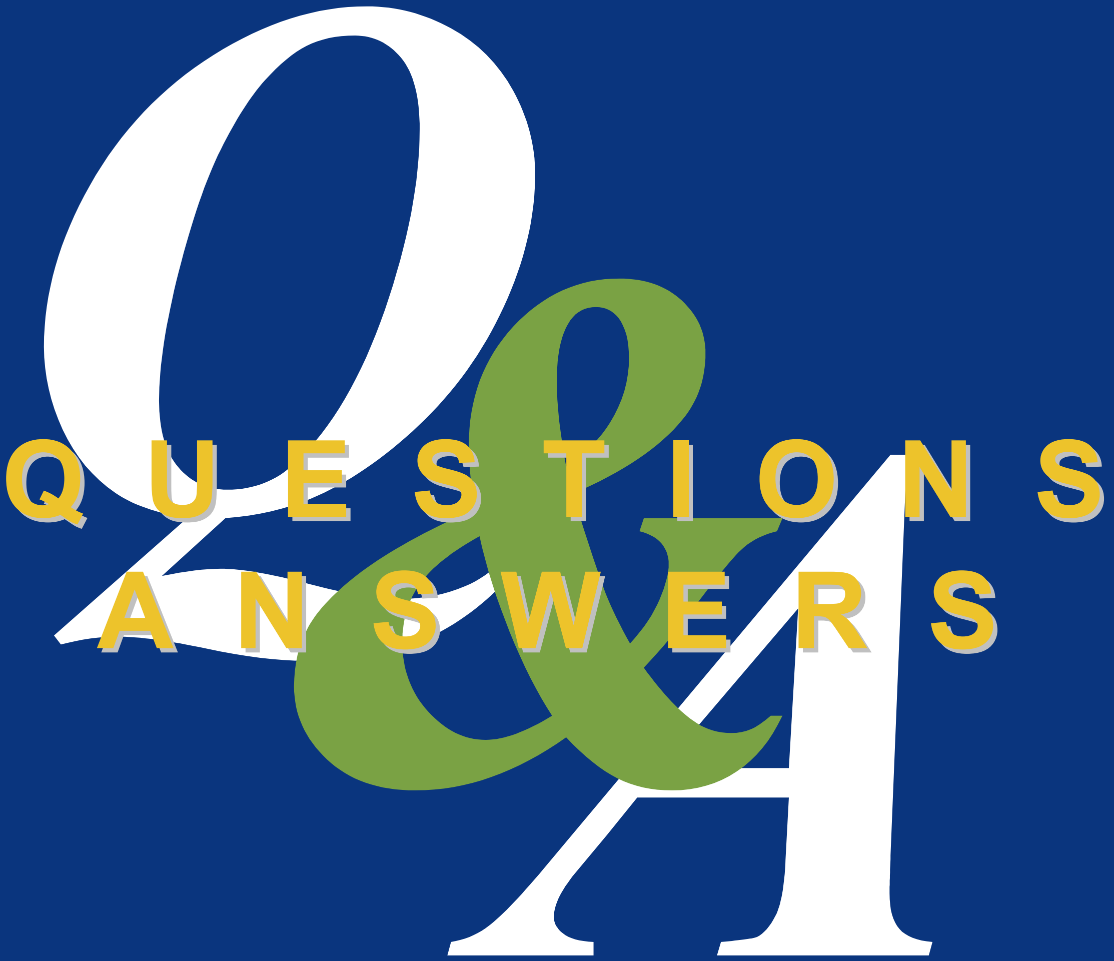
# Case Study - Conclusion

- Total load window before tuning was over 8 hours
- Inventory load would take around 70 mins

- Total load window after tuning is 5 hours and 30 mins
- Inventory load takes 30 mins

# Conclusion

- Wait event based performance analysis is superior to ratio based method
- Simplifies problem solving for most complex Oracle Configurations
- Parallel Query option should be exercised after careful analysis and sizing
- Data distribution plays critical role in PQO and Data Warehouse



# QUESTIONS
# ANSWERS

# HP WORLD 2003

Solutions and Technology Conference & Expo

Interex, Encompass and HP bring you a powerful new HP World.

interex
shared knowledge • shared power

encompass
AN HP USER GROUP

hp
invent