



IBM Solutions Technical Brief

**Diagnosing Oracle® Database Performance on AIX®
Using IBM® NMON and Oracle Statspack Reports**

**Dale Martin
IBM Oracle Solutions Team
Advanced Technical Support**

**Version: 1.0
Date: October 31, 2006**

1. INTRODUCTION.....	4
2. SPECIAL NOTICES	4
3. TRADEMARKS.....	5
4. FEEDBACK	5
5. NMON OVERVIEW	6
5.1. <i>Installation</i>	6
5.2. <i>Collecting NMON data</i>	7
5.3. <i>Running the NMON Analyser</i>	7
6. ORACLE STATSPACK OVERVIEW	8
6.1. <i>Installation</i>	8
6.2. <i>Collecting Statspack data</i>	8
6.3. <i>Generating a Statspack report</i>	9
7. COLLECTING THE DATA FOR REVIEW.....	11
8. ANALYZING NMON DATA	13
8.1. <i>Server Configuration Information</i>	13
8.1.1. <i>The AAA Worksheet</i>	13
8.1.2. <i>The BBBB Worksheet.....</i>	14
8.1.3. <i>The BBBC Worksheet.....</i>	14
8.1.4. <i>The BBBD Worksheet</i>	15
8.1.5. <i>The BBBL Worksheet</i>	15
8.1.6. <i>The BBBN Worksheet.....</i>	16
8.1.7. <i>The BBBP Worksheet.....</i>	16
8.1.8. <i>The BBBVG Worksheet.....</i>	17
8.2. <i>Where's the bottleneck?.....</i>	18
8.2.1. <i>CPU.....</i>	18
8.2.2. <i>Memory</i>	24
8.2.3. <i>I/O</i>	31
8.2.4. <i>Network.....</i>	42
9. ANALYZING ORACLE STATSPACK DATA.....	45
9.1. <i>Statspack Report Summary</i>	45
9.2. <i>The "Load Profile" Section.....</i>	46
9.3. <i>The "Instance Efficiency Percentages" Section.....</i>	47
9.4. <i>The "Top 5 Timed Events" Section</i>	47
9.5. <i>The "Cluster Statistics" Section</i>	51
9.6. <i>The "GES Statistics" Section.....</i>	53
9.7. <i>The "Wait Events" Section</i>	54
9.8. <i>The "Background Wait Events" Section.....</i>	55



9.9.	<i>The “SQL ordered by Gets” Section</i>	56
9.10.	<i>The “SQL ordered by Reads” Section</i>	58
9.11.	<i>The “Instance Activity Stats” Section</i>	59
9.12.	<i>The “Tablespace IO Stats” Section</i>	60
9.13.	<i>The “File IO Stats” Section</i>	62
9.14.	<i>The “Buffer Pool Advisory”</i>	63
9.15.	<i>The “PGA Memory Advisory” Section</i>	65
9.16.	<i>The “Enqueue Activity” Section</i>	66
9.17.	<i>The “Top 10 Buffer Busy Waits per Segment” Section</i>	66
9.18.	<i>The “Top 10 Row Lock Waits per Segment” Section</i>	67
9.19.	<i>The “Shared Pool Advisory” Section</i>	68
9.20.	<i>The “SGA Memory Summary” Section</i>	69
9.21.	<i>The “SGA breakdown difference” Section</i>	70
9.22.	<i>The “init.ora Parameters” Section</i>	70
10.	AWR REPORT EXAMPLES	72
10.1.	<i>The “Report Summary” Section</i>	72
10.2.	<i>The “Load Profile” Section</i>	72
10.3.	<i>The “Instance Efficiency Percentages” Section</i>	73
10.4.	<i>The “Top 5 Timed Events” Section</i>	73
10.5.	<i>The “Cluster Statistics” Section</i>	74
10.6.	<i>The “GES Statistics” Section</i>	75
10.7.	<i>The “Wait Events” Section</i>	75
10.8.	<i>The “Background Wait Events” Section</i>	76
10.9.	<i>The “SQL ordered by Gets” Section</i>	76
10.10.	<i>The “SQL ordered by Reads” Section</i>	76
10.11.	<i>The “Instance Activity Stats” Section</i>	77
10.12.	<i>The “Tablespace IO Stats” Section</i>	77
10.13.	<i>The “File IO Stats” Section</i>	78
10.14.	<i>The “Buffer Pool Advisory” Section</i>	79
10.15.	<i>The “PGA Memory Advisory” Section</i>	80
10.16.	<i>The “Enqueue Activity” Section</i>	80
10.17.	<i>The “Shared Pool Advisory” Section</i>	81
10.18.	<i>The “SGA Memory Summary” Section</i>	81
10.19.	<i>The “SGA breakdown difference” Section</i>	82
10.20.	<i>The “init.ora Parameters” Section</i>	82
11.	CONCLUSIONS	83
12.	REFERENCES	83
13.	ACKNOWLEDGEMENTS	83



1. Introduction

For Oracle Database environments, ongoing performance monitoring and tuning activities are essential to getting the most of your systems investment. In order to be most effective, monitoring and tuning must be performed at multiple levels of the solution stack. This includes the hardware, Operating System, Database and Application levels.

This paper discusses two commonly available performance monitoring tools – Oracle Statspack and IBM NMON – and how they can be used to monitor and analyze possible performance issues for Oracle Database applications running in an AIX/System p™ server environment. In Oracle10g environments, Automatic Workload Repository (AWR) reports can be used in place of Statspack. A number of real-world examples of Statspack and NMON data are provided.

This paper assumes that the reader is familiar with Oracle Database as well as the AIX/System p server environment.

2. Special Notices

Copyright 2006 IBM Corporation. All Rights Reserved.

Neither this documentation nor any part of it may be copied or reproduced in any form or by any means or translated into another language, without the prior consent of the IBM Corporation.

The information contained in this document is subject to change without any notice. IBM reserves the right to make any such changes without obligation to notify any person of such revision or changes. IBM makes no commitment to keep the information contained herein up to date.

The information contained in this document has not been submitted to any formal IBM test and is distributed on an "as is" basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Performance data contained in this document was determined in a controlled environment; therefore the results which may be obtained in other operating environments may vary significantly. No commitment as to your ability to obtain comparable results is any way intended or made by this release of information.

Version 1.0, published October 31, 2006



3. Trademarks

AIX, IBM, System p and System x are trademarks or registered trademarks of the International Business Machines Corporation.

Oracle is a registered trademark of Oracle Corporation.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

4. Feedback

Please send comments or suggestions for changes to dalem@us.ibm.com

5. NMON Overview

NMON (Nigel's Monitor) is written and maintained by Nigel Griffiths from the IBM Technical Sales Support team in the UK. NMON automatically collects a wide range of data, including CPU utilization, memory usage, paging activity, I/O activity, and much more. NMON versions are available for AIX as well as Linux® and can be used to analyze system level performance in System p and System x environments.

The “NMON performance” component may either be run interactively to monitor real time performance information, or in the background to collect data to a text file for later analysis. In this whitepaper, we will be focusing on the background data collection method.

The “NMON Analyser” is a complementary tool developed by Stephen Atkins (in the IBM Advanced Technical Sales (ATS) UK organization). It is written in VBA for Excel and displays NMON data in graphical format.

NMON and the NMON analyser are **NOT OFFICIALLY SUPPORTED** by IBM, but may be downloaded for “AS IS” customer use from the IBM Wiki Web site at:

<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmon> and
<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmonanalyser>.

For more details on installation and use of NMON or NMON analyser, please refer to the documentation that comes with NMON or available on the Wiki sites identified above. A summary of the installation process is provided here, but be sure to consult the latest NMON documentation to ensure you have current installation details.

5.1. Installation

An AIX system admin can typically install NMON in just a few minutes:

- FTP download the nmon4aixYYY.tar.gz file (where YYY is the current version number) from the Wiki Web site (see above) to your server.
- Unzip the file:

```
gunzip nmon4aixYYY.tar.gz
```

- Untar the file:

```
tar xvf nmon4aixYYY.tar
```

- If not already done, enable the collection of disk statistics (iostat) by executing the following as root:

```
chdev -l sys0 -a iostat=true
```

- A zip file containing NMON Analyser tool can be downloaded to your PC from the IBM Wiki Web site at:
<http://www.ibm.com/collaboration/wiki/display/WikiPtype/nmonanalyser>. The downloaded file should then be unzipped.

5.2. Collecting NMON data

In order to direct NMON data to a file, invoke it with the “-f” flag. This will create an output file in the current directory called “<hostname>_date_time.nmon”. Other commonly used flags for background data collection are as follows:

Flag	Description
-help	Provides a description of the options.
-c <number>	The number of snapshots to be taken before the NMON collector run terminates.
-s <seconds>	The time interval (in seconds) between snap shots.
-F <filename>	Can be used in place of the “-f” flag to provide a user supplied file name.
-r <runname>	Specifies the “runname” to be placed in the data file. This defaults to hostname.
-T	Includes top processes and command line arguments information in the output file.
-A	Includes asynch I/O (aio) server information in the output file.
-x	Provides some default options (-fdt -s300 -c96) for general capacity planning purposes. These options provide 15 minute intervals for 24 hours.
-V	Includes Volume Group specific information.
-g <filename>	Specifies user defined groups of disks.
-m <dir name>	Specifies which directory the output file is to be written to (defaults to current local directory).

5.3. Running the NMON Analyser

- If you have collected a large amount of NMON data (> 64K lines), you should presort the NMON output file. The NMON Analyser will stop and return an error message if it encounters an unsorted file that contains more than 64K lines:

```
sort -A mymachine_311205_1030.nmon > mymachine_311205_1030.sorted.nmon
```

- FTP the .nmon (or .sorted.nmon) file to your PC using the ASCII or TEXT options.
- Open the NMON Analyser spreadsheet.
- Optionally, change any of the Analyser settings.
- Click on the “Analyse nmon data” button and select the file(s) to be processed.

For more details, refer to the “NMON_Analyser User Guide” which comes with the analyser tool.

6. Oracle Statspack Overview

Per Metalink note # 149115.1, “What is Statspack and where are the READMEs?”, “Oracle Statspack is a set of SQL, PL/SQL and SQL*Plus scripts which facilitate the collection, automation, storage and viewing of Oracle performance data”. These scripts are used to collect performance data snapshots from the Oracle V\$ performance views, store the snapshot data in a set of tables and generate performance reports from that snapshot data.

Beginning in Oracle 10g, Automatic Workload Repository (AWR) reports are available that provide similar information as Statspack. Over time, AWR reports will continue to be enhanced. If you have a 10g environment, you might prefer to use AWR reports instead of Statspack reports. Statspack reports are provided as examples in this paper, because they are available in Oracle 9i as well as 10g environments.

6.1. Installation

If you are using The Oracle 10g Enterprise Manager (EM), it is possible to integrate the Statspack utility with EM to allow Statspack information to be viewed directly from the EM console. For details on how to do this, see Oracle Metalink note # 274436.1, “How to Integrate Statspack with EM 10G”.

If Oracle Enterprise Manager 10g is not being used, Oracle Metalink note # 149113.1, “Installing and Configuring Statspack Package” provides details on how to install Statspack. Below is an example of the interactive installation procedure:

```
Sqlplus
SQL> connect / as sysdba
SQL> @?/rdbms/admin/spcreate
```

The spcreate script will create a PERFSTAT user and create a number of database objects, including tables, constraints and packages. The script will prompt for the name of the default and temporary tablespaces for the PERFSTAT user. The SYSTEM tablespace should not be used for these. Typically, a TOOLS tablespace would be used for PERFSTAT’s default tablespace and the instance’s TEMPORARY tablespace used for PERFSTAT’s temporary tablespace.

6.2. Collecting Statspack data

Oracle Metalink note # 149121.1, “Gathering a Statspack snapshot” provides instructions on how to setup Statspack to collect snapshot data.

Before collecting any snapshot data, ensure that the Oracle initialization parameter `timed_statistics` is set to `TRUE`. This can be done at instance startup time by setting `timed_statistics=TRUE` in the `init.ora` or `spfile`. It can also be changed dynamically using the ‘alter system’ command.

The snapshot level parameter controls the amount and type of data collected by Statspack. For general analysis purposes, the snapshot level should be set to at least 5. If more detailed data is desired, the snapshot level can be set higher.

- Level 6 provides additional SQL plans and SQL Plan Usage statistics
- Level 7 provides additional Segment level statistics
- Level 10 provides additional Parent and Child Latch statistics (Level 10 should be used cautiously, since it can have a significant performance impact)

Snapshots can either be taken interactively, or they can be scheduled to be run periodically. Taking snapshots interactively can be useful in test environments where you only want to capture information for a particular test period. In that case, you would take one snapshot at the beginning of the test run and another snapshot at the end of the test run (with zero or more intermediate snapshots during the middle of the test run).

Below is an example of how a snapshot can be taken interactively:

```
Sqlplus
SQL> connect perfstat/perfstat
SQL> execute statspack.snap;
```

In production environments, it is typical to schedule snapshots to automatically run periodically. This provides the ability to make comparisons of Oracle performance over time. Snapshots can be automatically scheduled using the Oracle `dbms_job` procedure, or an AIX level utility such as `'cron'`.

In order to use `dbms_job`, the Oracle initialization parameter `"job_queue_processes"` must be set to 1 (or higher). Oracle supplies a default `'spauto.sql'` script which will schedule snapshots to be done once an hour on the hour. This script can be modified to meet particular customer requirements. A 15 minute snapshot frequency is fairly typical for production environments.

6.3. Generating a Statspack report

Oracle Metalink note # 149124.1, "Creating a Statspack performance report" provides instructions on how to create Statspack performance reports from Statspack snapshot data.

To create a Statspack report, execute the `'spreport.sql'` script while connected as the `PERFSTAT` user. The script will prompt for the beginning snapshot id, the ending snapshot id and the name of the report text file to be produced. Take care when choosing the beginning and ending snapshot ids; Statspack reports are not valid if the instance has been shutdown/restarted within the Statspack reporting period.

Below is an example of how the spreport.sql script is invoked:

```
sqlplus
SQL> connect perfstat/perfstat
SQL> @?/rdbms/admin/spreport.sql;
```

Note: Oracle10g users might prefer to use the new Automatic Workload Repository (AWR) feature. AWR provides reporting capability similar to Statspack, plus some additional information not available in Statspack. For more information about AWR, see Metalink note # 276103.1, “Performance Tuning Using 10g Advisors and Manageability Features”.

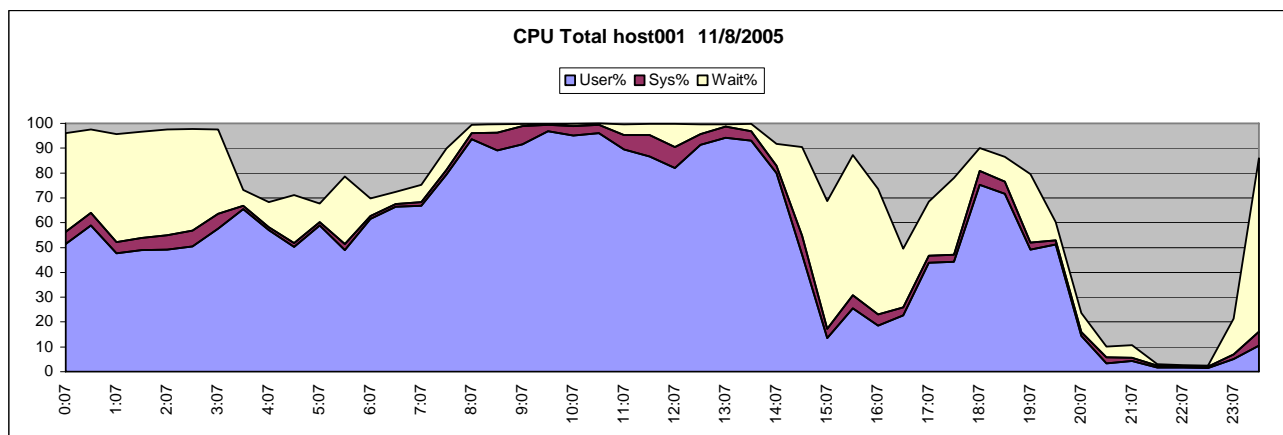
7. Collecting the Data for Review

Before collecting data for review, appropriate monitoring period(s) and snapshot duration(s) to be used for data collection should be determined. So that the NMON and Statspack data can be correlated to each other, it is important that the collection of these two sets of data is coordinated. In other words, they should both be run at the same time and should use compatible sets of snapshot intervals and other parameters.

For ongoing performance monitoring, or an initial “health check”, it is fairly typical to collect up to 24 hours of NMON data (perhaps scheduled to begin at midnight) at a time, using a 15 minute snapshot interval. It is typically a good idea to set the NMON snapshot interval in relation to the total NMON collection period. For example, if there are more than a couple hundred snapshots in a single collection period, the resulting file can get rather large and may be more difficult to analyze. On the other hand, if there are only a few snapshot intervals during the collection period (e.g less than 10), it may not provide a very good picture of performance trends and workload variability during the collection period.

For Oracle Statspack, it is fairly common to automatically collect statistics every 15 to 60 minutes on a scheduled basis. Then, when desired, one or more Statspack reports can be generated for time periods of interest. Some customers automatically generate a single Statspack report for each 24-hour interval (day). While this may be useful for high level trend information, using such a large reporting interval tends to mask issues that might appear during certain times of day (for example, during prime shift, or while running a particular batch job). Using a fairly frequent snapshot frequency (for example, once every 15 minutes) provides the option to zero in on particular time periods by generating a Statspack report for a single (or at least a small number of) snapshot intervals.

Below is a sample of the CPU utilization summary for a 24-hour NMON data collection. As can be seen from the chart, CPU utilization was very high from about 8:00 am to 2:00 pm and there was a brief upward spike in CPU utilization around 6:00 pm. In order to better understand what was going on during these periods of high CPU activity, we might want to begin by reviewing Statspack reports (for example) for the 8:00 - 8:30, the 10:00 – 10:30 and the 18:00 – 18:30 time period.





In those situations where there are known performance issues that need to be diagnosed, the monitoring period(s) and snapshot frequency should be adjusted to the workload being analyzed. For example, if you are trying to analyze performance of a batch job that runs for 15 minutes every day, you might want to collect NMON data for a total of 45 minutes (15 minutes before, 15 minutes during and 15 minutes after) with a fairly short snapshot interval (for example, 15 seconds). And, you may want to use a fairly short Statspack snapshot interval, for example, once every 5 minutes to provide the ability to generate multiple Statspack reports before, during or after the job run. Or, you might want to manually generate a new snapshot just before and just after the batch job runs.

When reasonably large snapshot intervals are chosen, NMON and Statspack both tend to generate fairly light system overhead. However, monitoring overhead is directly proportional to the snapshot interval, so caution should be used when considering extremely short snapshot intervals (for example, NMON < 15 seconds, Statspack < 5 minutes).

In situations where the system is very CPU bound and many disks are attached to the system or you have a large number of processes running and utilize the “-T” option in NMON, the time stamps generated by NMON for each interval need to be validated. If the time difference between two consecutive snapshots is larger than the specified capture interval, then the capture interval needs to be extended. A significant error in CPU utilization and other counters can be observed if NMON is not able to capture / process the performance data well within the specified data capture period, especially in micro partitioned LPARs.

8. Analyzing NMON data

8.1. Server Configuration Information

At the start of the collection run, NMON collects a substantial amount of server configuration information. This initial data collection can take several minutes, particularly in environments with a large number (>75) of disk devices. The NMON Analyser then creates a number of spreadsheet worksheets where this information can be reviewed. These worksheets can be a good place to start, particularly if you're not intimately aware of the specific hardware configuration, AIX release levels, or AIX parameter tuning that has been done.

8.1.1. The AAA Worksheet

Provides high level information about the AIX version/release levels and server hardware configuration:

AIX	5.2.0.77
build	AIX52 /usr/local/bin/nmon_aix52ml5 -F /var/tmp/nmon.host200.2006-05-09-12PM
command	-s 300 -c 144 -t
cpus	32
date	9-May-06
disks_per_line	150
hardware	Architecture PowerPC Implementation RS64-IV or POWER4 64 bit
Host	host200
interval	300 HW-type=CHRP=Common H/W Reference Platform Bus=PCI
Kernel	LPAR=Dynamic Multi-Processor 64 bit
ML	5
progname	nmon_aix52ml5
runname	host200
snapshots	48 @(#)nmon v11e - AIX52(\$Id: nmon11.c,v 1.10 2006/04/05 15:18:14 nag
subversion	Exp \$)
Time	12:01.0
User	root
Version	v11e
analyser	V3.1.0

8.1.2. The BBBB Worksheet

Provides a high level summary of the hdisks attached to the server LPAR:

Name	size(GB)	disc attach type
hdisk3	36.4	SCSI
hdisk2	146.8	SCSI
hdisk7	36.4	SCSI
hdisk6	146.8	SCSI
hdisk0	36.4	SCSI
hdisk1	36.4	SCSI
hdisk4	36.4	SCSI
hdisk5	36.4	SCSI
hdisk624	unknown	Hitachi-HDS
hdisk625	unknown	Hitachi-HDS
hdisk626	unknown	Hitachi-HDS
hdisk627	unknown	Hitachi-HDS
hdisk628	unknown	Hitachi-HDS
...

8.1.3. The BBBC Worksheet

Provides a high level summary of the hdisk to Logical Volume (LV) to file system mapping. The hdisk to Volume Group (VG) mapping and type of VG is also shown:

```

hdisk3:
LV NAME                LPs    PPs    DISTRIBUTION          MOUNT POINT
hd6                    135    135    27..27..27..27..27  N/A

hdisk2:
LV NAME                LPs    PPs    DISTRIBUTION          MOUNT POINT
hd5                    1      1      01..00..00..00..00  N/A
hd7                    72     72     00..72..00..00..00  N/A
hd4                    1      1      00..00..01..00..00  /
hd2                    7      7      00..00..07..00..00  /usr
hd9var                 8      8      00..00..08..00..00  /var
hd3                    8      8      00..00..08..00..00  /tmp
hd1                    8      8      00..00..08..00..00  /home

hdisk7:
LV NAME                LPs    PPs    DISTRIBUTION          MOUNT POINT
paginglv02            135    135    27..27..27..27..27  N/A
...
hdisk0                00cb41fe8b91860c    rootvg    active
hdisk2                002fe30f31f8fcf9    oravg    active
hdisk13               00cb41fea922890f    rac_data_cvg    concurrent
hdisk14               00cb41fea92289be    rac_data_cvg    concurrent
...

```

8.1.4. The BBBD Worksheet

Provides a high level summary of the I/O adapters installed:

Disk Adapter Information host200

Adapter_number	Name	Disks	Description
0	fcs0	198	FC Adapter
1	fcs1	198	FC Adapter
2	fcs2	198	FC Adapter
3	fcs3	198	FC Adapter
4	scsi2	1	Wide/Ultra-3 SCSI I/O Controller
5	scsi1	2	Wide/Ultra-3 SCSI I/O Controller
6	scsi5	2	Wide/Ultra-3 SCSI I/O Controller
7	scsi0	2	Wide/Ultra-3 SCSI I/O Controller
8	scsi4	2	Wide/Ultra-3 SCSI I/O Controller

8.1.5. The BBBL Worksheet

Provides LPAR configuration details when LPARs are being used. This information can be particularly useful when doing micro-partitioning with shared processor pools.

lparno	2
lparname	host720
CPU in sys	8
Virtual CPU	8
Logical CPU	16
Pool CPU	8
smt	2
capped	0
min Virtual	1
max Virtual	32
min Logical	1
max Logical	64
min Capacity	0.8
max Capacity	8
Entitled Capacity	6.4
Weight	224
min Memory	8192
max Memory	32768
online Memory	32768
Flags	LPARed DRable SMT-bound Shared UnCapped

8.1.6. The BBBN Worksheet

Provides a high level summary of the network adapters installed:

NetworkName	MTU	Mbits	Name
en2	1500	1024	Standard Ethernet Network Interface
en5	1500	1024	Standard Ethernet Network Interface
lo0	16896	0	Loopback Network Interface

8.1.7. The BBBP Worksheet

Provides a wealth of information about AIX kernel parameter settings, including vmo, no, ioo, and Workload Manager (WLM) related settings. If no data is displayed for vmo, ioo or no, then NMON was not run under the root user id.

When doing a general health check, this is an excellent place to look at to determine if general AIX for Oracle tuning “best practices” have been implemented, or if there are any unusual non-default AIX settings. Basic best practices tuning information is available from several sources, including the “Oracle Architecture and Performance Tuning on AIX” whitepaper available at:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP100657>

For example, following is some sample ‘vmo’ output from the **BBBP** worksheet:

vmo	NAME	CUR	DEF	BOOT	MIN	MAX	UNIT
vmo	DEPENDENCIES						
vmo	...						
vmo	lru_file_repage	1	1	1	0	1	boolean
vmo	-----						
vmo	lru_poll_interval	0	0	0	0	60000	
vmo	-----						
vmo	maxclient%	10	80	80	1	100	%
vmo	maxperm%						
vmo	-----						
vmo	maxfree	1536	128	128	16	200K	4KB
vmo	Minfree						
vmo	memory_frames						
vmo	-----						
vmo	maxperm	1257K		1257K			
vmo	-----						
vmo	maxperm%	10	80	80	1	100	%
vmo	...						

8.1.8. The BBBVG Worksheet

Provides high level information regarding the Volume Groups defined and the number of Physical Volumes per Volume Group. This worksheet is only provided when the NMON -V option is used.

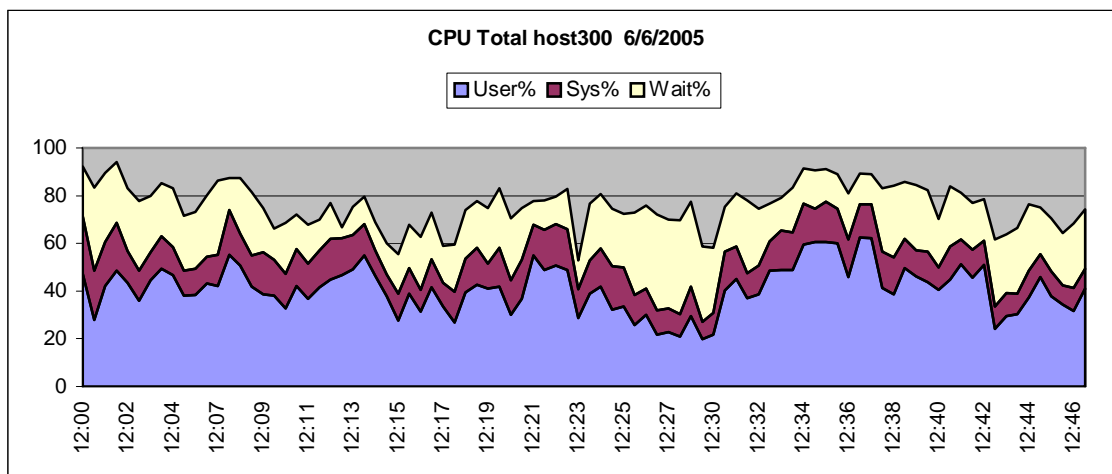
Volume Group Name	Number of Disks
rootvg	2
vglocalapp	4
None	3
vgsharedapps	8
vgwbbprd	21
vgdbexport	1
vgoemagent	1

8.2. Where's the bottleneck?

When analyzing potential performance issues, it is very helpful to understand whether or not there are any obvious Operating System or hardware level constraints or bottlenecks that are limiting overall throughput or performance. In most cases, system level bottlenecks (if any) will be CPU, Memory, I/O or Network related. Therefore, when reviewing NMON data, it is usually a good idea to start with a quick review to check for potential bottlenecks in each of these four areas.

8.2.1. CPU

The **CPU_ALL** worksheet provides CPU utilization statistics for the NMON collection period. The following chart shows an LPAR using dedicated processors, which has moderately active CPU utilization (Peak User% + Sys% < 75%). There is no evidence of a CPU hardware capacity issue.

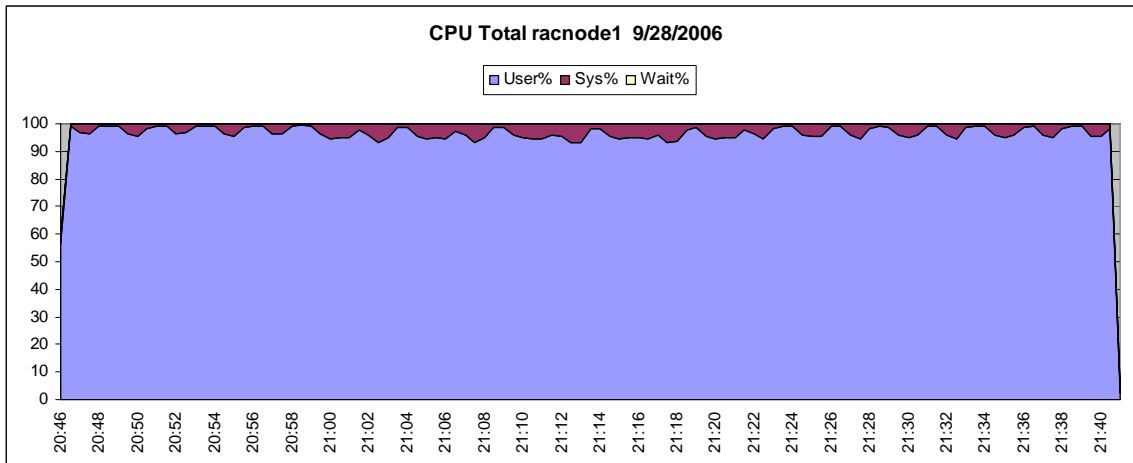


Note that the chart also shows Wait%. The Wait% statistic is the percentage of time that a processor was idle (nothing on the run queue) and there was at least one outstanding I/O outstanding started on that processor. This statistic is often misinterpreted or given more emphasis than it deserves. A high Wait% value is not necessarily an indication of an I/O performance issue. In many cases, high Wait% values simply indicate that the workload is relatively I/O bound. In other words, the application does a lot of I/O compared to the amount of computational processing that is done. As overall server demand increases (and CPU utilization increases proportionally), the reported Wait% often declines. (See the “CPU Total” chart below for an example of this.)

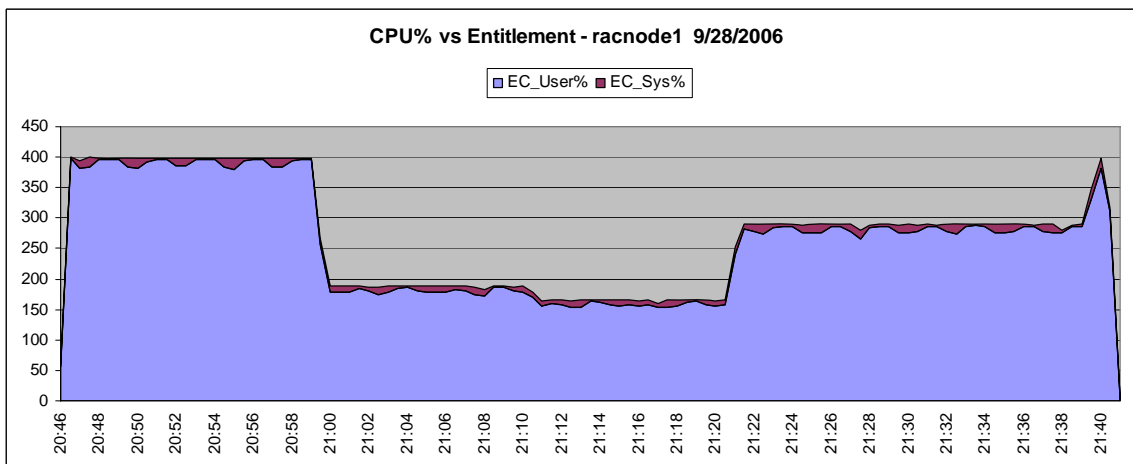
Therefore, at best, a high Wait% value is an indication that I/O subsystem performance should be reviewed further. By itself, it is never a definitive indication of I/O related issues. As will be discussed later, some of the other information provided by NMON and Oracle Statspack reports provide a far better indicator of I/O subsystem performance.



When uncapped shared processor partitions (micropartitions) are being used, the percentages shown on the utilization chart in the CPU_ALL worksheet can be somewhat misleading. These percentages shown are based on actual usage vs. the total capacity available to the partition, including capacity in excess of that partition's entitled capacity. The following graph shows a workload that is using 100% of the CPU capacity that is made available to it.

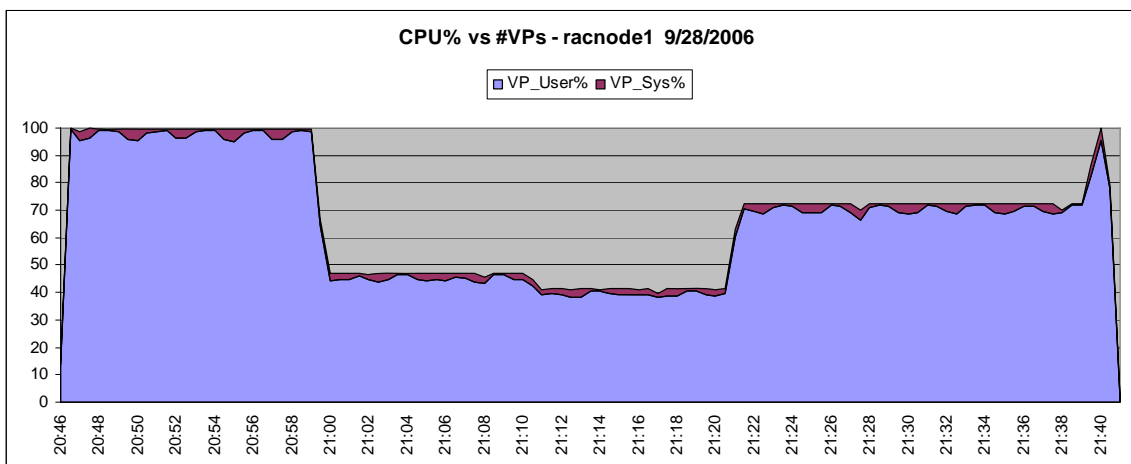


From the previous graph, it might appear that the partition's CPU usage is constant over time. However, it is not. The LPAR worksheet provides further information in this case. The following graph from the LPAR worksheet shows CPU utilization vs. the entitled capacity for that partition. As can be seen, the CPU usage varied between about 150% and 400% of entitled capacity, depending on how much excess CPU resource was available in the shared pool:

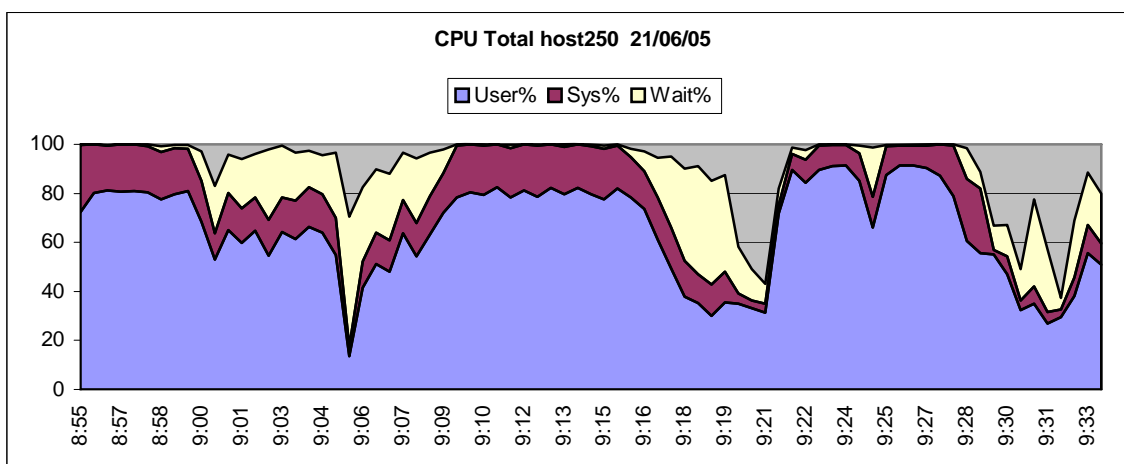




The LPAR worksheet also has a graph that shows CPU utilization as expressed as a percentage of the potential peak capacity based on the number of Virtual CPUs for that partition. In this particular example, there were 8 Virtual CPUs and entitled capacity of 2.0 CPUs. Therefore, if we are using 8 Virtual CPUs (left hand side of the graph) worth of CPU capacity the chart below shows 100% of the 8 Virtual CPUs of capacity being used, whereas, the chart above shows 400% of the 2.0 CPUs worth of entitled capacity being used – different views of the same workload:

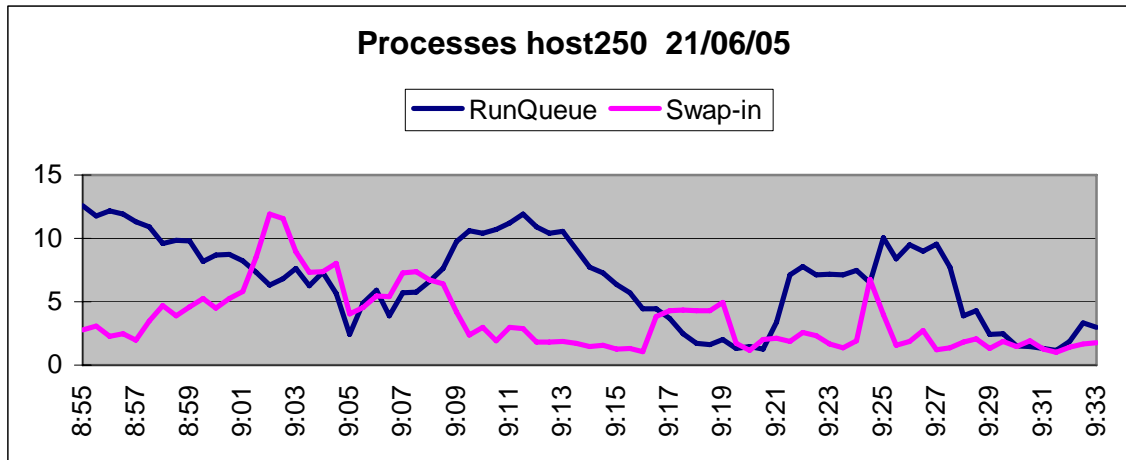


The following graph shows a system that is severely CPU bound. Note the long periods of time where the CPU (User% + Sys%) is pegged at 100% utilization. In Oracle DB environments, the Sys% is typically below 10-15%. Significantly higher Sys% values might be symptomatic of an underlying tuning issue, for example, excessive lru or syncd activity.

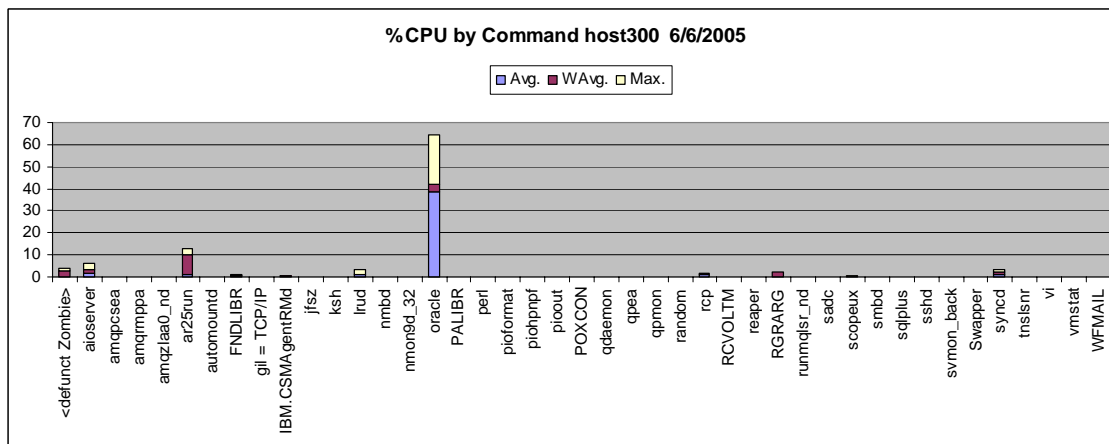




The **PROC** worksheet provides information about the system run queue. Notice how the peaks in CPU utilization tend to be associated with peaks in the number of processes on the run queue. Since this was a 4-way server, at times, the number of processes on the run queue averaged as much as 3 times the number of processors on the server. The Swap-in value indicates the number of processes that blocked waiting for an I/O to complete. A large number of blocked processes can potentially result from an I/O performance problem. However, like CPU WAIT%, the number of blocked processes is not necessarily the best indicator of I/O performance.



When trying to address high CPU utilization issues, it is important to understand what processes on the system are responsible for the bulk of the CPU usage. If NMON was run with the `-T` (or `-t`) option, the NMON Analyser will also produce a **TOP** worksheet which provides information about the processes who are the top consumers of CPU. Below is an example of the first chart from the **TOP** worksheet.



If Oracle is primarily responsible for the CPU usage (as is the case here), it is advisable to look into Oracle performance details to determine if any DB or application level tuning or design changes are warranted (for example, adding a more discriminating index to avoid a



table scan). An Oracle Statspack report is ordinarily the place to begin when analyzing these types of issues. If the high CPU utilization cannot be addressed through tuning or application redesign, additional (or faster) processors might be required to satisfy the workload CPU demand. If AIX system related processes, for example, aioserver, lrud or syncd show large amounts of CPU time, this may indicate the need for AIX level tuning.

In addition to the chart summary, the **TOP** worksheet also provides detailed information for each of the top processes per snapshot interval. For example, if we wanted to investigate the “ar25run” process that showed up on the chart above, we have the following detail:

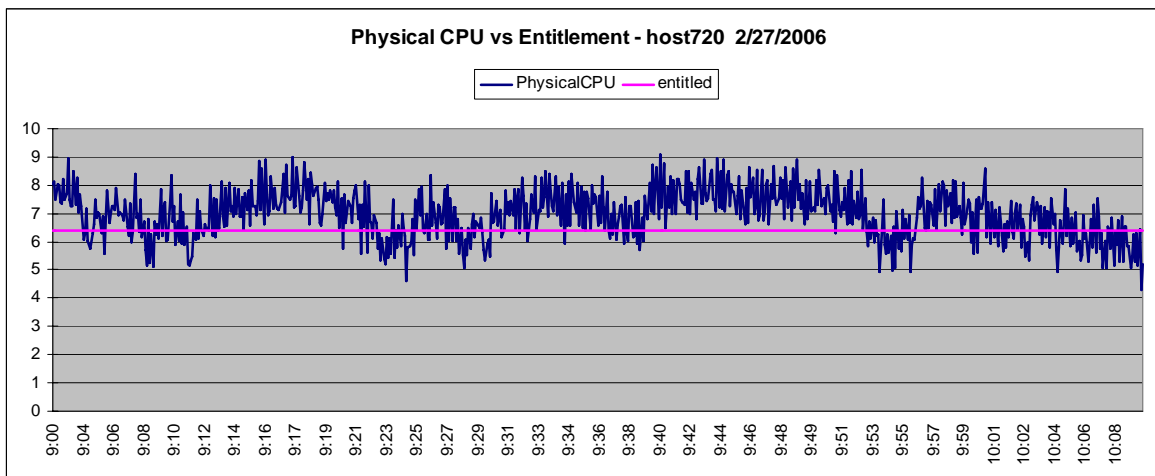
Time	PID	%CPU	%Usr	%Sys	Threads	Size	ResText	ResData	CharIO	%RAM	Paging
12:17:31	138352	28.5	27.1	1.4	1	19332	5900	6404	297140	0	14
12:18:02	138352	59.9	57.3	2.6	1	23176	5908	10248	1471960	0	31
12:18:33	138352	18.6	18.1	0.5	1	30388	5908	17460	38370	0	57
12:19:04	138352	16.4	15.9	0.5	1	36284	5908	23356	31226	0	47
12:19:35	138352	15.2	14.7	0.5	1	42064	5908	29136	30282	0	47
12:20:06	138352	21.3	16.9	4.4	1	49976	5908	37048	231048	0	64
12:20:38	138352	48	33.7	14.3	1	62052	5948	49124	618497	0	100
12:21:09	138352	99.9	90.4	9.5	1	62052	5948	49124	155759	0	38
12:21:41	138352	99.9	91.4	8.5	1	62052	5948	49124	163003	0	39
12:22:13	138352	101	90	11	1	62052	6052	49124	217896	0	53
12:22:46	138352	98.3	84.7	13.6	1	62052	6052	49124	266478	0	65
12:25:57	655346	0.1	0	0.1	1	16504	6192	3576	172	0	3

If we specified the NMON collector `-T` option (rather than the `-t` option), the NMON Analyser also produces an **UARG** worksheet. That worksheet provides command line argument detail for the processes shown on the **TOP** worksheet. Below is the **UARG** worksheet detail for the “ar25run” processes:

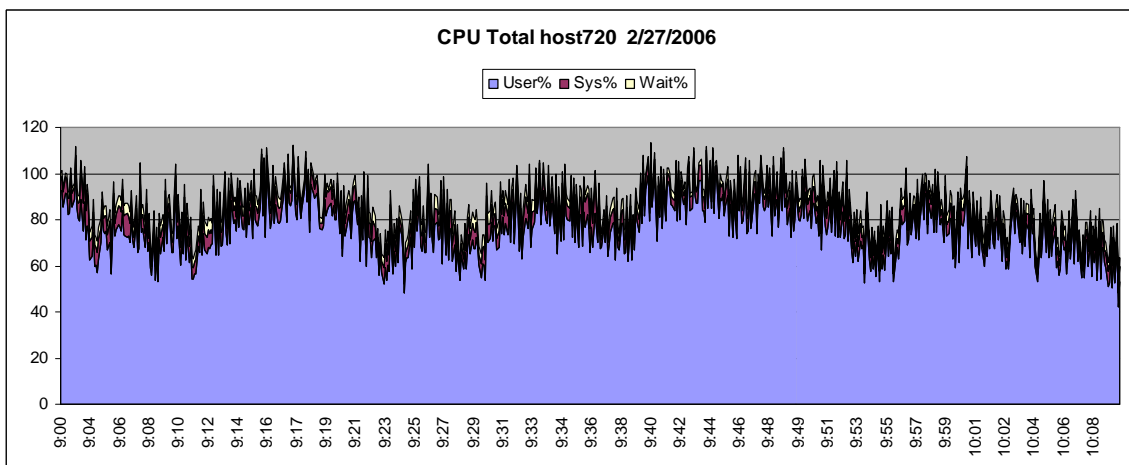
Time	PID	ProgName	FullCommand
12:17:31	138352	ar25run	ar25run P_CONC_REQUEST_ID=1848357 p_mode='I' p_actual_cost_flag='Y' p_revenue_flag='Y' p_budgets_flag='Y' p_commitment_flag='Y' p_debug_mode='Y' p_gen_rep='Y' report=/ora142/u001/app/applmgr/r11/PXXXX/pa/11.0.28/reports/PXYZ.rdf batch=yes PLSQL_DATE_FORMAT=DD-MON-RR destype=file desname=/apps1/u000/common/outPXXXX/VDEST.1848357 desformat=/ora142/u001/app/applmgr/r11/PXXXX/fnd/11.0.28/reports/HDT pagesize=132x60
12:25:57	655346	ar25run	ar25run P_CONC_REQUEST_ID=4207089 P_FILENAME='373306x305.csv' P_GL_DATE='06-JUN-2005' report=/ora142/u001/app/applmgr/r11/PXXXX/flap/1.1.1/reports/PAABC.rdf batch=yes PLSQL_DATE_FORMAT=DD-MON-RR destype=file desname=/apps1/u000/common/outPXXXX/LDEST.4207089 desformat=/ora142/u001/app/applmgr/r11/FLN0P/fnd/11.0.28/reports/HDT pagesize=132x60



When a shared processor partition is being used, there are a number of other worksheets that provide useful information. We have already covered the [BBBL](#) worksheet, which provides a summary of the LPAR configuration. The first chart from the **LPAR** worksheet shows physical CPU usage (in CPU processor units) vs. the LPAR entitlement. For uncapped LPARs, the physical CPU usage for an LPAR can exceed the entitlement for that LPAR as shown in the chart below:



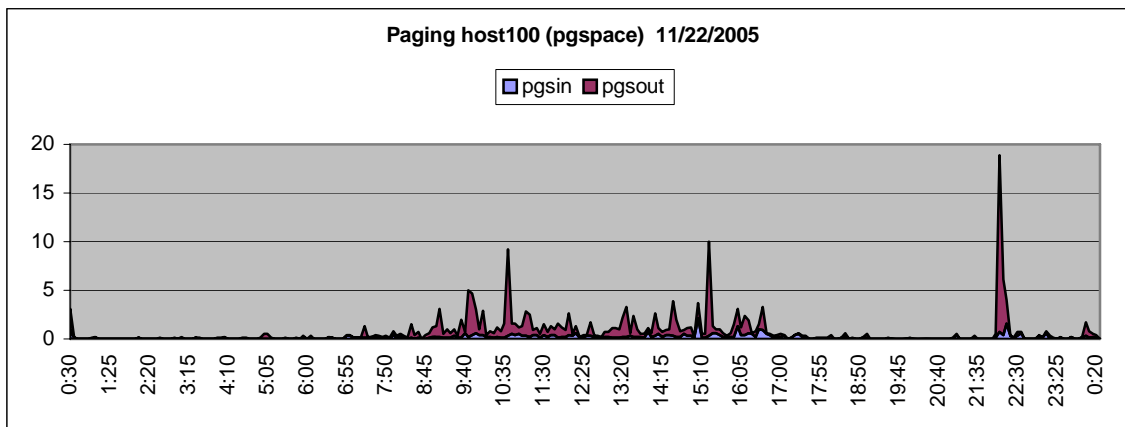
The following chart from the CPU_PHYSICAL worksheet provides a similar view, except that actual CPU utilization is shown as a percentage of entitled capacity. Note that the ratio of physical processor to entitlement (shown as %entc in the output of the `lparstat` command) will generally be higher than CPU% on the CPU_ALL sheet. The reason for this is that a partition that is within its entitlement may wait for a short period of time before ceding a processor that enters an I/O wait or becomes idle. This can eliminate unnecessary context switches.





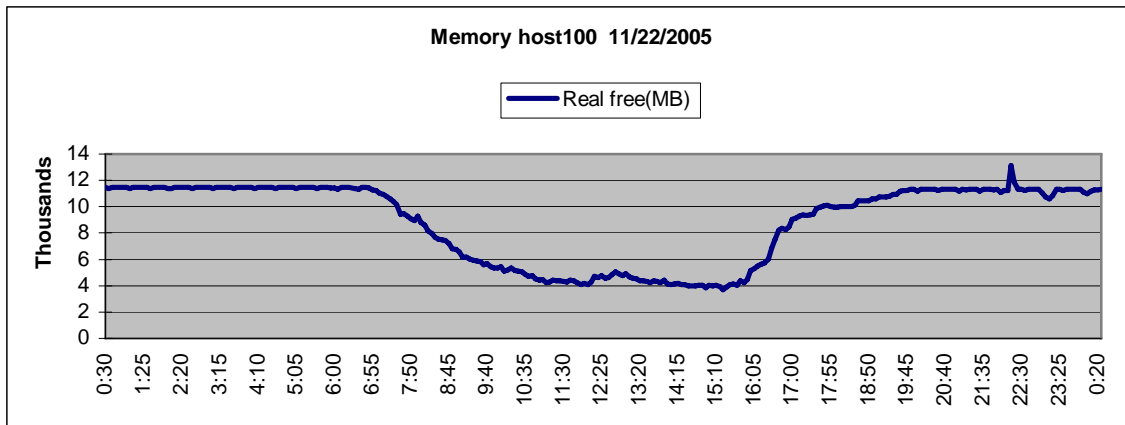
8.2.2. Memory

The **PAGE** worksheet provides a graphical summary of paging activity (both to the swap space as well as for file systems). In Oracle DB environments, we normally do not want to see any pgsin or pgsout activity to the system swap files. The following chart is an example of where paging to the system swap space is occurring:

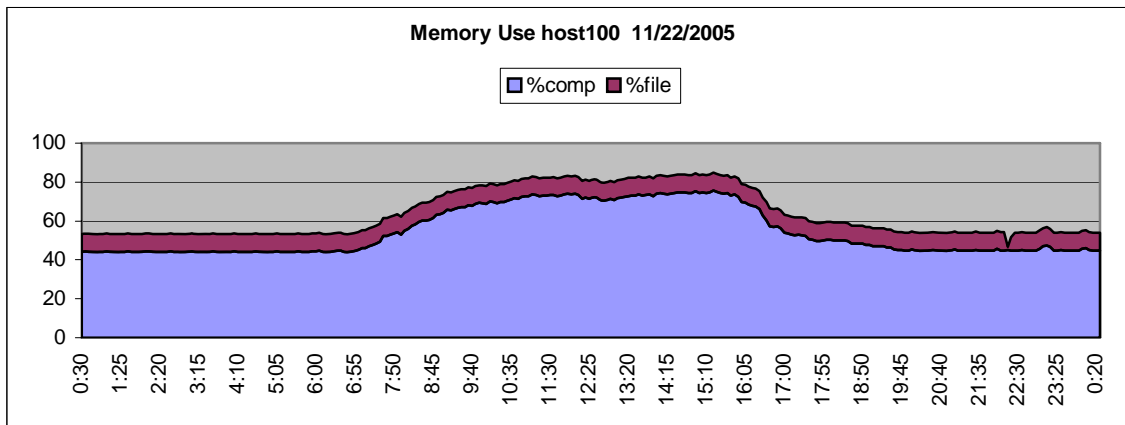




When we look at the **MEM** worksheet, we can see that while paging is occurring, there is still a significant amount of free physical memory available (at least 4 GB). This suggests that the paging activity can probably be eliminated through AIX Virtual Memory Manager (VMM) tuning.

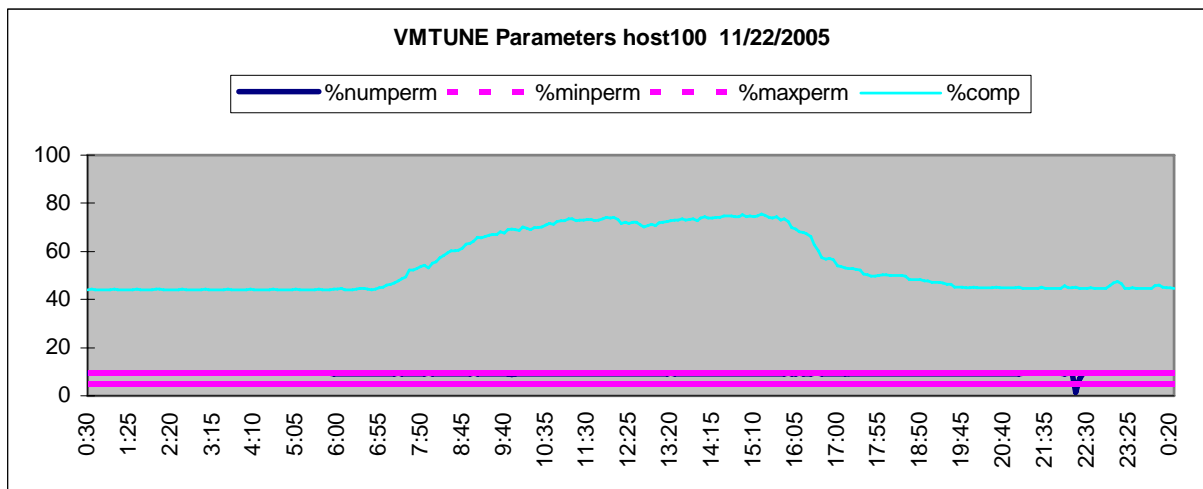


The **MEMUSE** worksheet provides additional information as to how the physical memory is being utilized. From the following chart, we can see that during the peak memory demand, slightly more than 70% of the total physical memory is being used for computational pages and about 10% of the physical memory is being used for filesystem pages. All of the empty (gray) area above the burgundy colored %file component is available free memory.





The **MEMUSE** worksheet also provides a graphical display of some of the key VMTUNE (vmo) parameters.



From the chart above, we can see that minperm% is set to 5 and maxperm% is set to 10. These settings can be confirmed by reviewing the vmstat or vmo output in the [BBBP](#) worksheet. While it is not shown on the graph, we can also confirm that maxclient% is set to 10 and lru_file_repage=1. We are not using all of the available free memory for file system cache because a hard limit has been set (strict_maxperm=1 or strict_maxclient=1).

```

vmstat -v          5.0 minperm percentage
vmstat -v          10.0 maxperm percentage
vmstat -v          9.7 numperm percentage
vmstat -v          586191 file pages
vmstat -v          0.0 compressed percentage
vmstat -v          0 compressed pages
vmstat -v          9.9 numclient percentage
vmstat -v          10.0 maxclient percentage
vmstat -v          595905 client pages
    
```

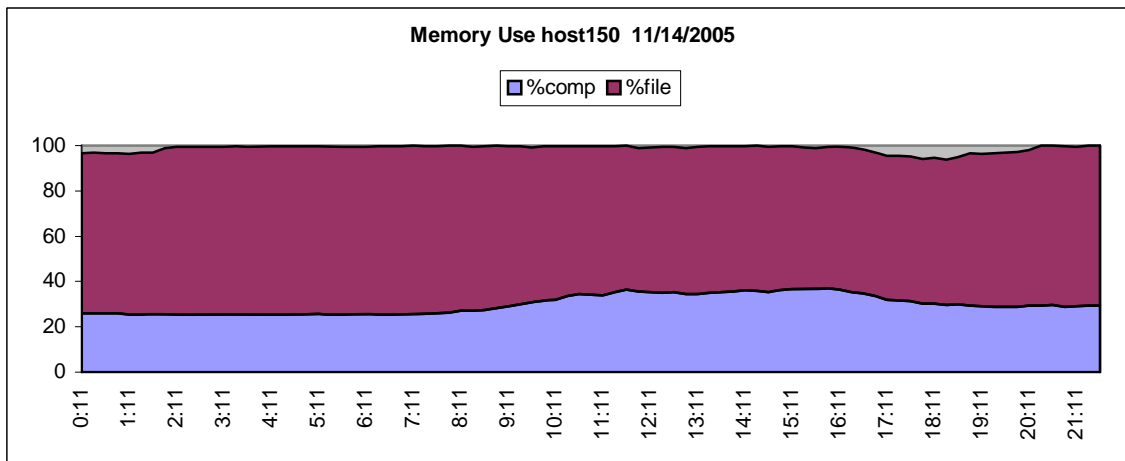
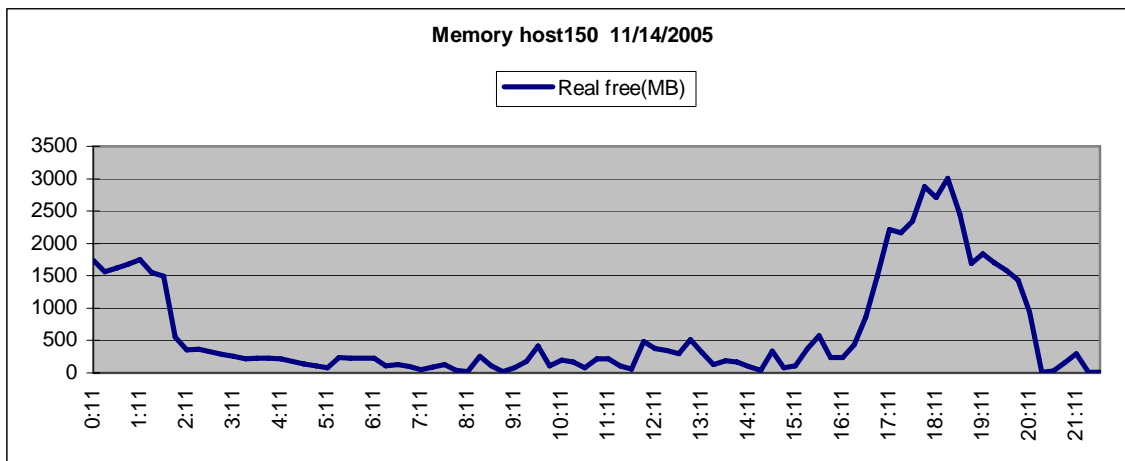
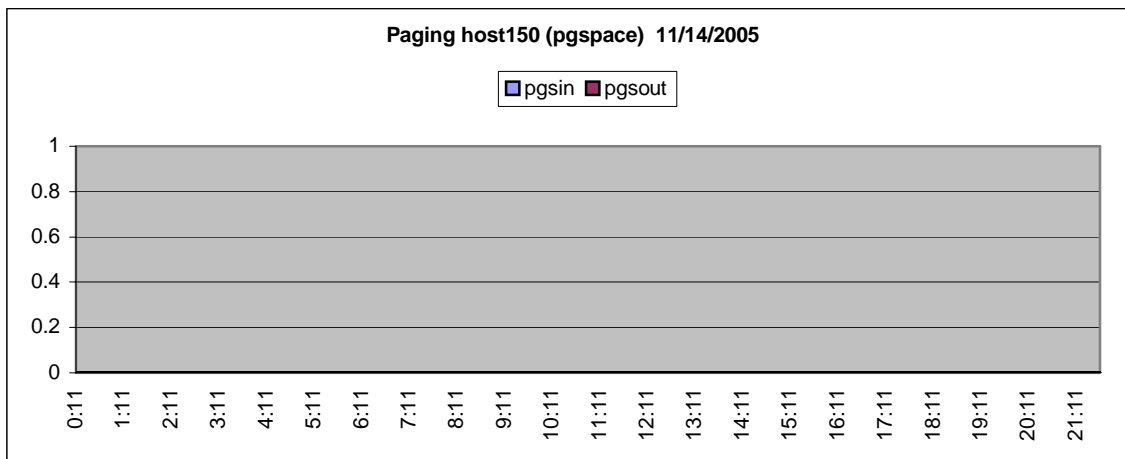
```

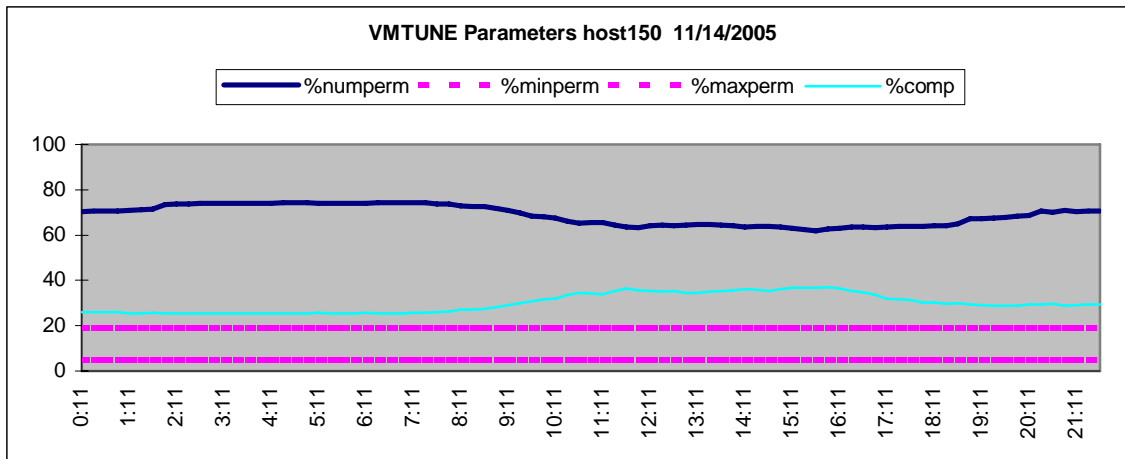
vmo -L  lru_file_repage      1      1      1      0      1
    
```

In this particular situation, the server has multiple memory pools defined. Paging is probably occurring because one or more of the pools was running short of free memory (and the numperm% for that particular pool was below maxperm%). This is not evident in the system level memory statistics because some of the other pools still have significant amounts of free memory available or higher numperm% within that individual pool. In this case, setting setting lru_file_repage=0 can eliminate the paging activity. At current AIX 5.2 and 5.3 release levels, lru_file_repage=0 is a generally accepted tuning practice. For AIX 5.3, we also generally recommend that maxperm%, maxclient%, strict_maxperm and strict_maxclient all be left at their default values unless there are specific indications that a change is warranted.



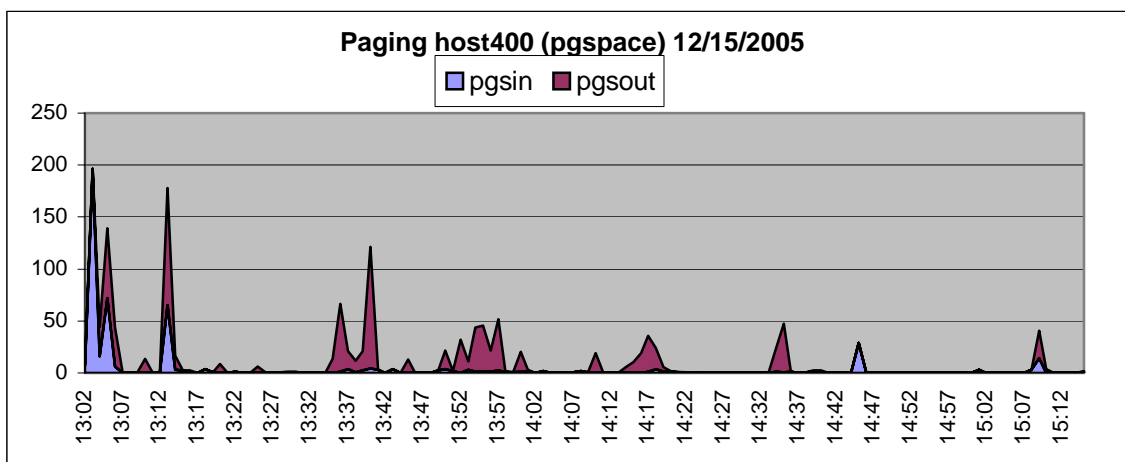
The following charts show a situation where AIX is properly tuned. Even though there is very little free memory available much of the time, there is absolutely no paging activity. This is because when a new memory page is required, AIX is always stealing file system pages, rather than computational pages.

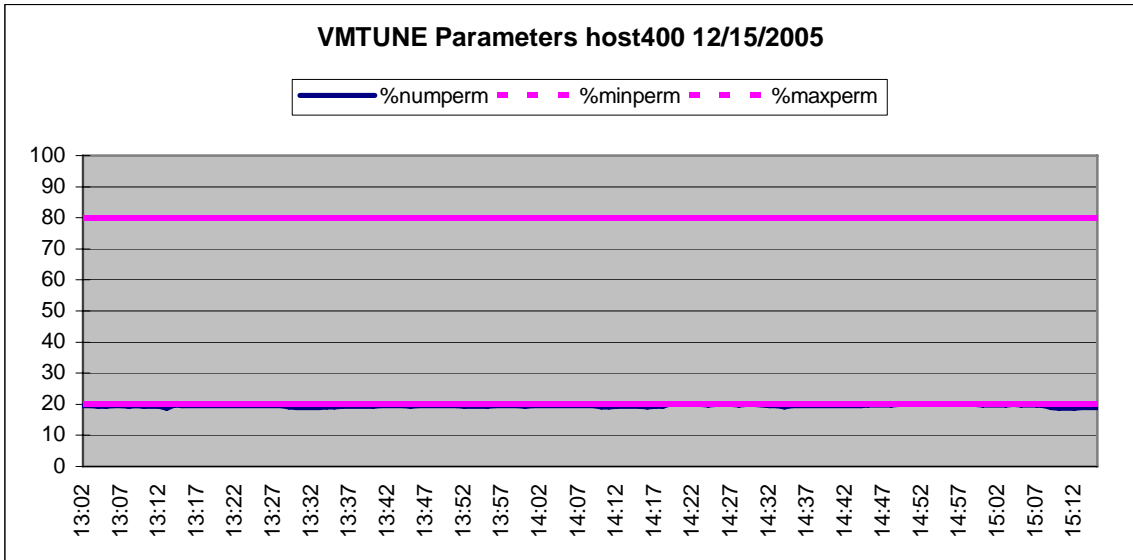
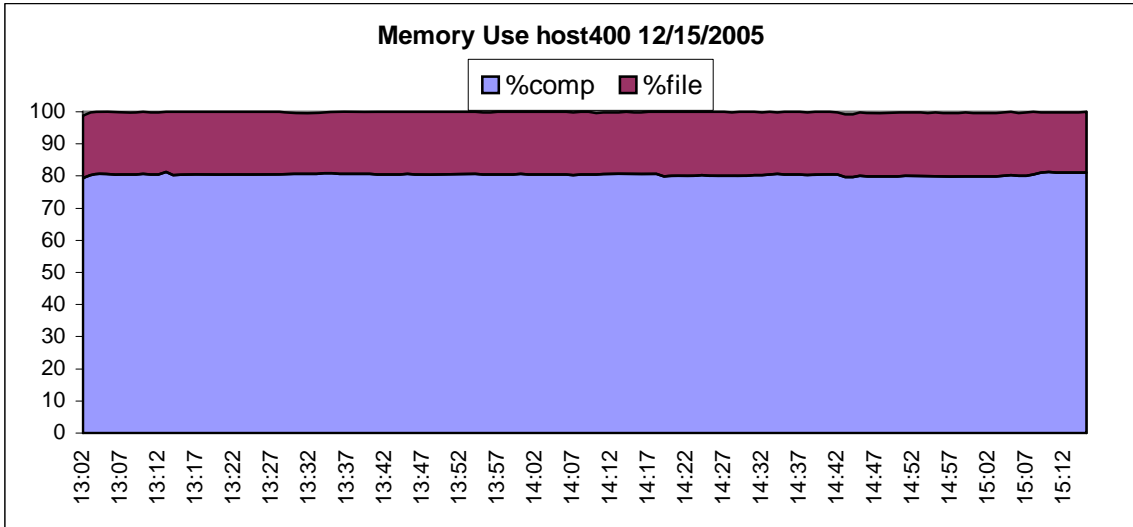
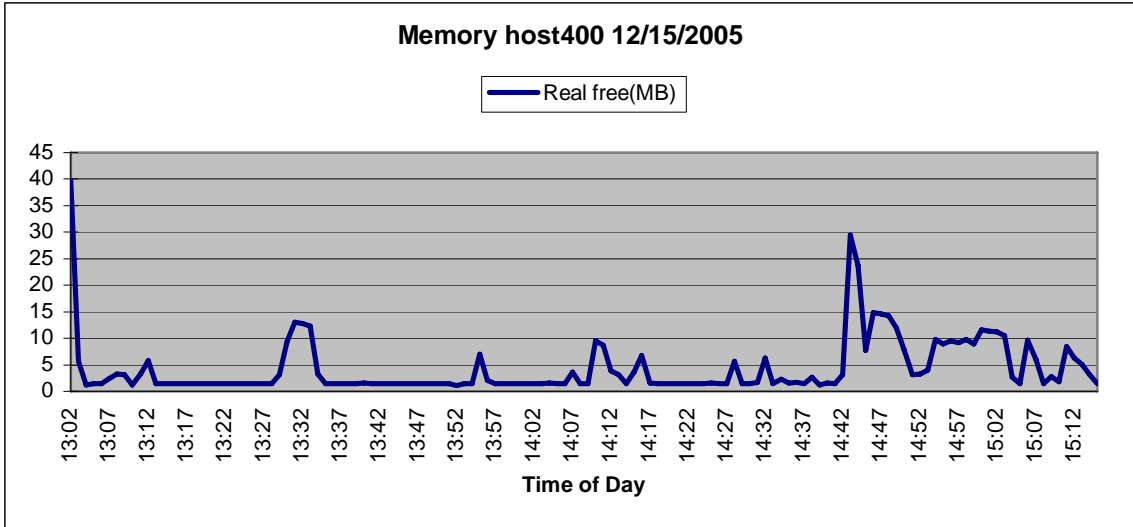




Note that in this case, all of the memory not required for computational page use is being used for file system cache (numperm% > maxperm%). This means that AIX is using a soft maxperm% (strict_maxperm=0) or maxclient% (strict_maxclient=0) limit.

The following set of charts shows an environment where system paging is occurring because vmo tuning has not been performed:





Note that `minperm%` and `maxperm%` are both at their default settings (20% for `minperm%` and 80% for `maxperm%`) and that `numperm% <= maxperm%`. In this situation, the first thing that needs to be done is reduce the `minperm%` setting.

Whenever the AIX `%numperm` value (file system pages as percent of total physical memory) is less than the `vmo minperm%` setting and AIX needs to free up additional memory, it will always page out a computational page. In Oracle DB environments, we typically set the `minperm%` parameter to a relatively low value (`<= 5`) so that we can maximize the amount of physical memory that is potentially available for computational pages.

However, if the virtual memory requirements for computational pages exceed the amount of physical memory available for computational pages (for example, 95% if `minperm%=5`), this is going to result in system paging. If the `minperm%` value has already been tuned, the ways to address the paging issue would be to reduce the overall computational page demand (such as reducing the Oracle SGA or PGA memory footprint, or reducing the number of concurrent users or other processes) or increase the amount of physical memory in the server.

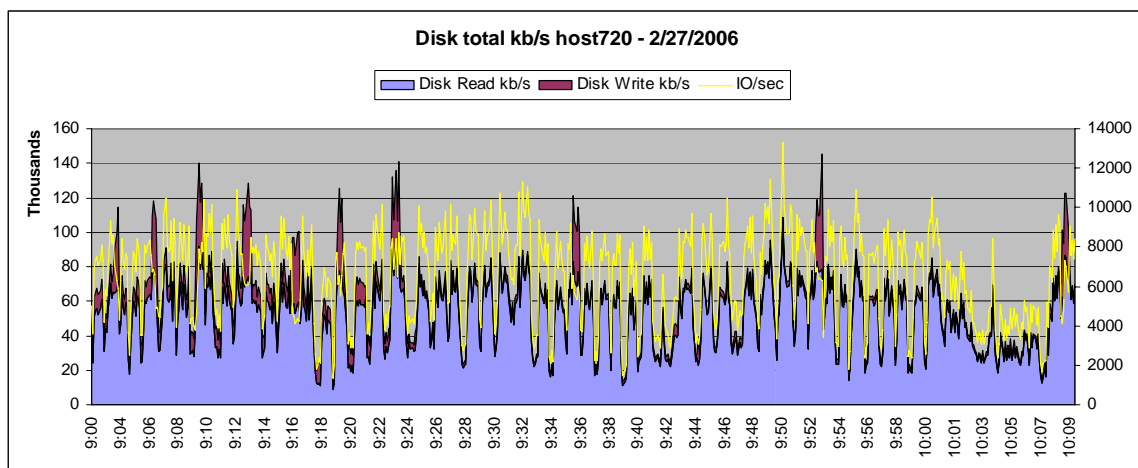
8.2.3. I/O

The **SYS_SUMM** worksheet provides high level System I/O statistics over the entire NMON collection period.

Total System I/O Statistics

Avg tps during an interval:	6,525
Max tps during an interval:	13,275
Max tps interval time:	9:50:20
Total number of Mbytes read:	188,012
Total number of Mbytes written:	26,053
Read/Write Ratio:	7.2

The **DISK_SUMM** worksheet provides a high level summary of the overall system I/O activity. This can be a good place to start to get an initial feel of the I/O workload. What is the read to write ratio? Is the I/O activity relatively uniform over time or somewhat erratic? Does the I/O workload tend to increase in conjunction with an increase in CPU consumption, or does CPU utilization decrease during periods of high I/O demand?



NMON for AIX does not provide reliable information about I/O response times as the reported “estimated” I/O response times are calculated by the Analyser rather than actually captured from AIX. In some cases, the estimated values might be significantly lower than actual. The Oracle Statspack report provides information about I/O response times (particularly for read I/Os) as perceived by Oracle. Beginning with AIX 5.3 maintenance level 03, the iostat command (-D option) can report detailed information about I/O response time and throughput for physical disks. The AIX filemon trace facility can also be used if detailed I/O response time statistics are required at the File, Logical Volume or Physical Volume levels. Filemon traces are fairly resource intensive and should only be run as required and for relatively short periods of time.

However, NMON does provide a number of useful I/O related statistics. In general, we are looking for an I/O subsystem where the I/O activity is well-balanced across all of the



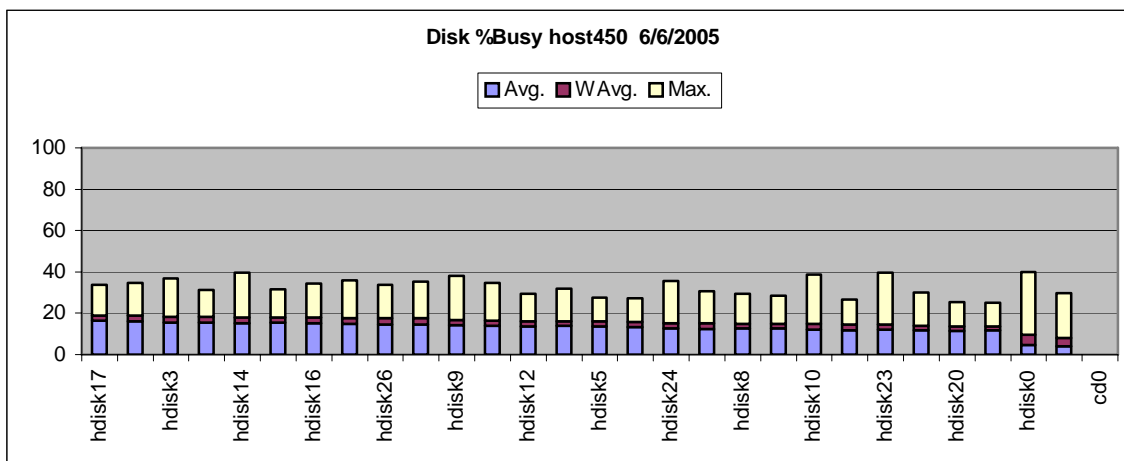
available disks and where peak device utilization is moderate (for example, < 30% if the hdisk corresponds to a single physical drive, <60% if the hdisk is RAID based). If these things are true, the resulting I/O response times will typically be acceptable.

GPFS or Oracle Automatic Storage Management (ASM) based implementations tend to exhibit good I/O balance because they both implement data “striping” techniques internally. It is also possible to implement various forms of “striping” at the AIX Logical Volume Manager (LVM) level when conventional JFS or JFS2 filesystems are used, or for Raw Devices implementations. For details on a data layout strategy for achieving well-balanced I/O utilization, see the whitepaper available at:

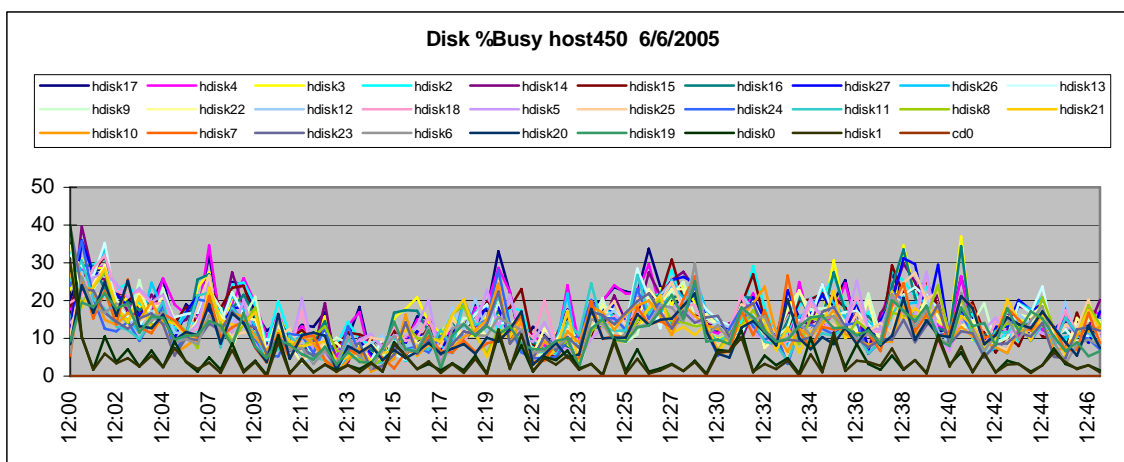
<http://www-03.ibm.com/support/techdocs/atmastr.nsf/WebIndex/WP100319>



The following charts show an environment where a good data layout policy has resulted in relatively balanced I/O activity across the available disks and relatively moderate device utilization. The first two charts are from the **DISKBUSY** worksheet and show device utilization to be fairly consistent across devices. “Avg” is the average percent busy across the entire NMON collection period, including snapshots where the device was 0% busy. “WAvg” is the average percent busy for those snapshots where the device was > 0% busy. “Max” is the highest percent busy record for any of the snapshots.

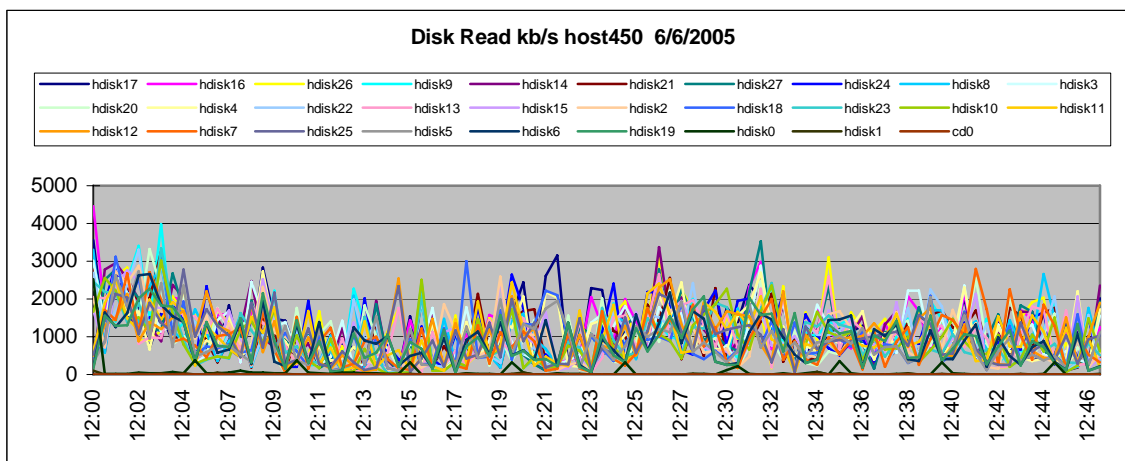
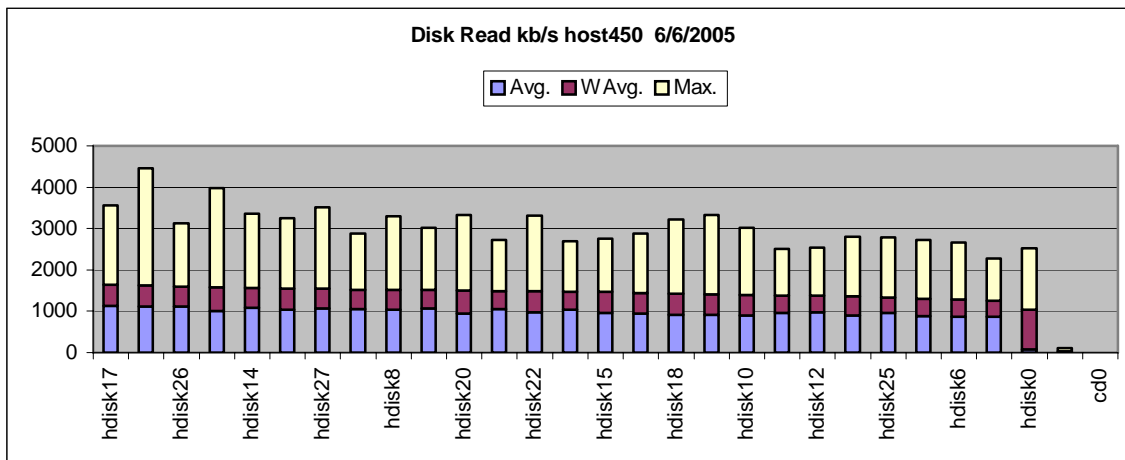


While the bar chart above provides an indication of how well-balanced the I/O activity was “on average” over the entire NMON collection period, there is also a graph which shows the actual %busy over time for every device. This chart can be used to see what the I/O distribution is over time and whether or not the same (or different) devices tend to be busy at the same times.



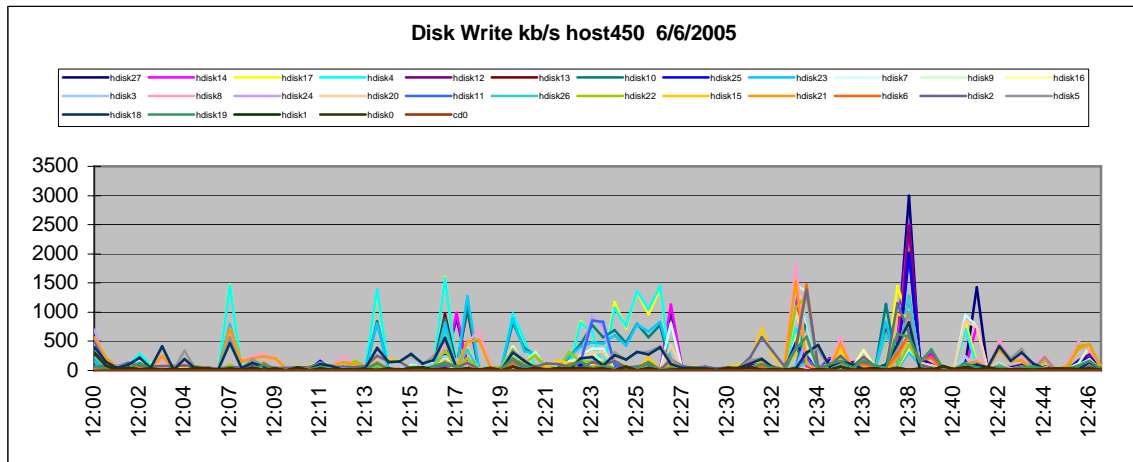
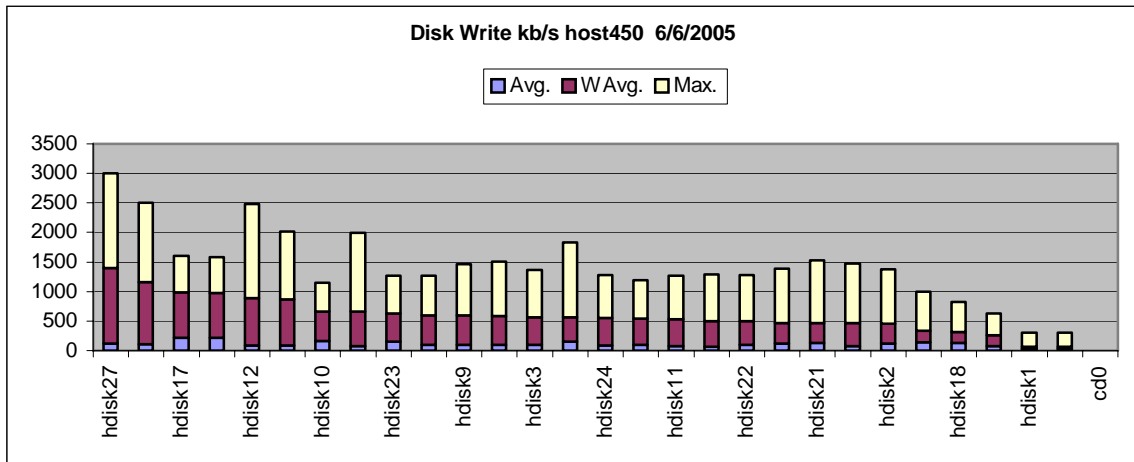


The next two charts from the **DISKREAD** worksheet show that read activity is fairly well-balanced across the available disks:



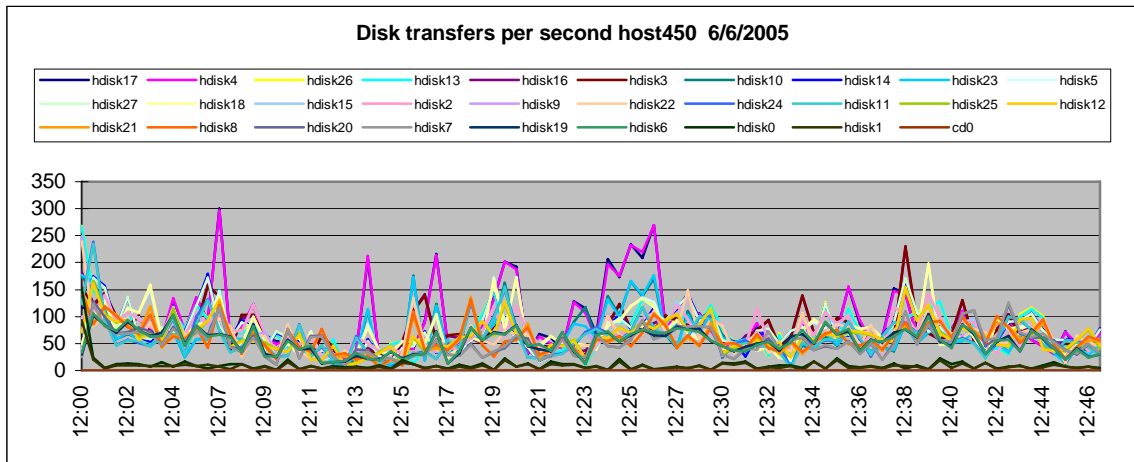
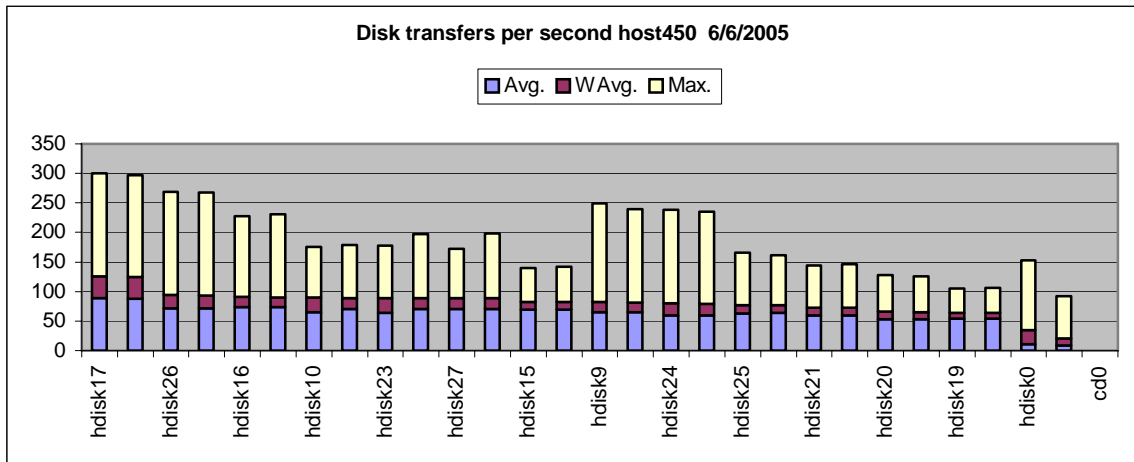


The next two charts from the **DISKWRITE** worksheet show that write activity is fairly well-balanced across the available disks:



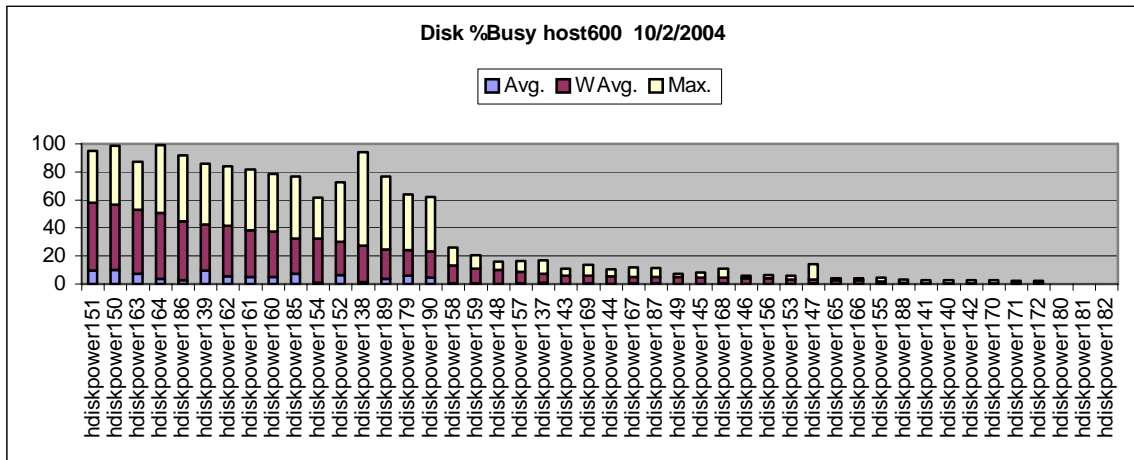


And, the final two charts from the **DISKXFER** worksheet show that total read and write I/O activity is fairly well-balanced across the available disks:

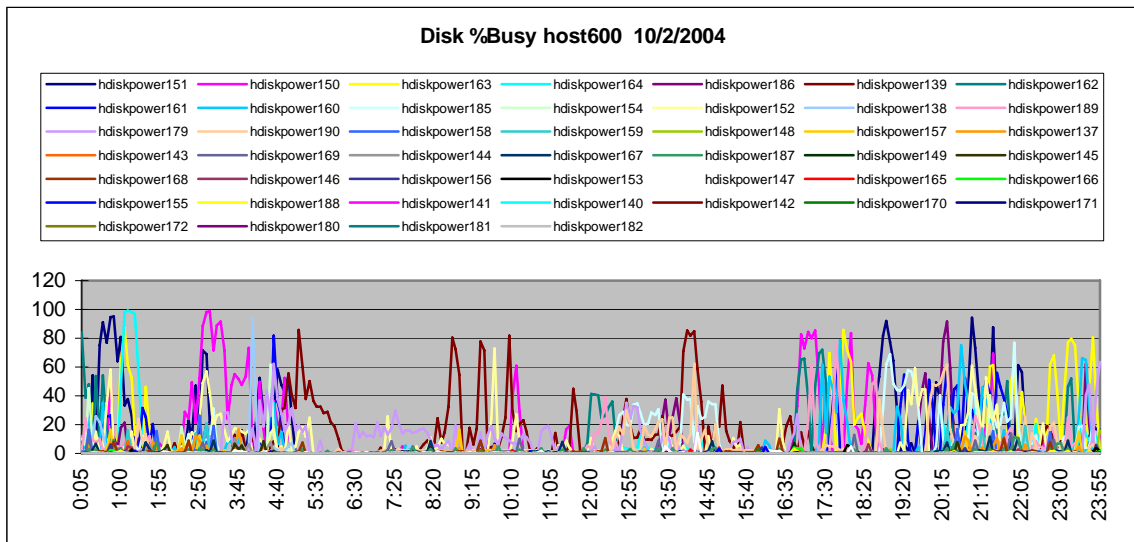




The following charts are from an Oracle environment where there is no disciplined data layout strategy and the I/O workload is not well-balanced across the available disks. The first two charts are from the **EMCBUSY** worksheet. As can be seen, there are a number of Powerpath devices with peak device utilization near 100% (sustained over a 5 minute snapshot interval). At the same time, many other devices are nearly idle. In many situations, this leads to significantly higher I/O response times on the heavily active devices than would be possible if the I/O activity were more evenly balanced.

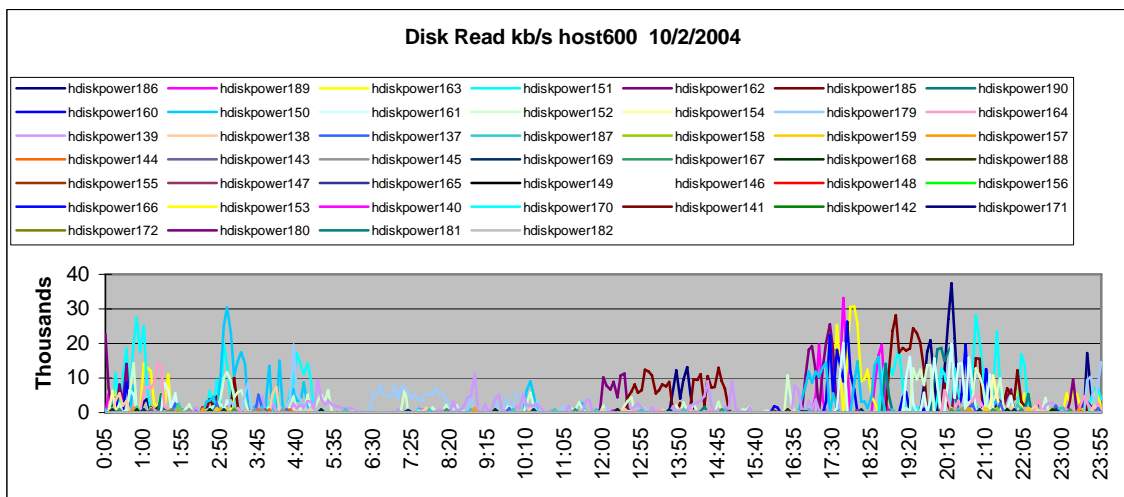
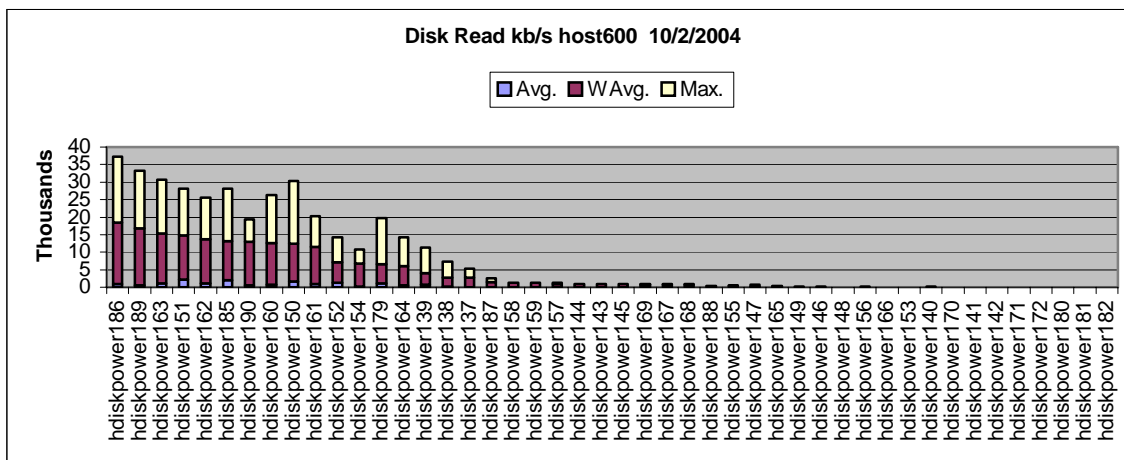


However, when detailed device utilization is viewed over time, only one or two of the Powerpath devices show high device utilization at any given time. This means that the total I/O throughput capacity of the application is being gated by the throughput capacity of just one or two Powerpath devices.

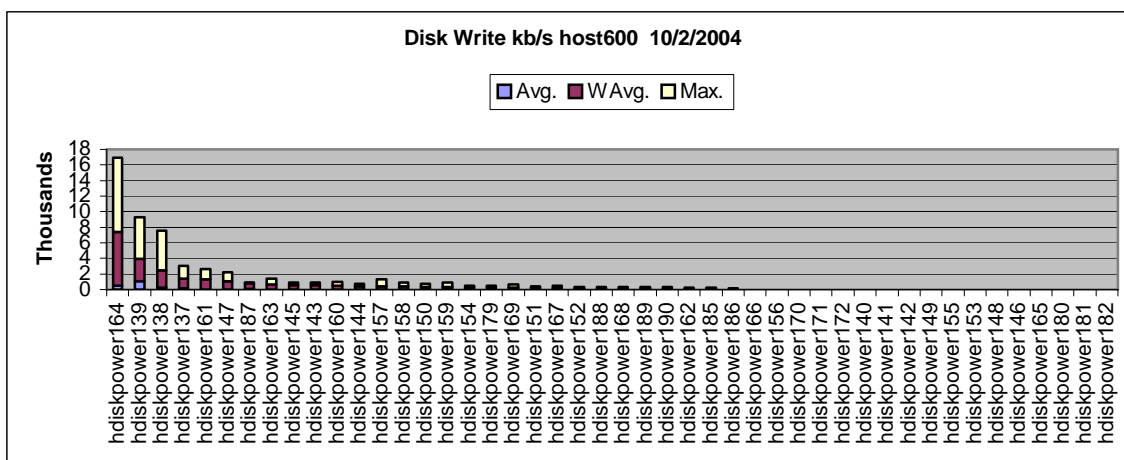




Similar skewing can be seen in the Read I/O statistics from the **EMCREAD** worksheet:

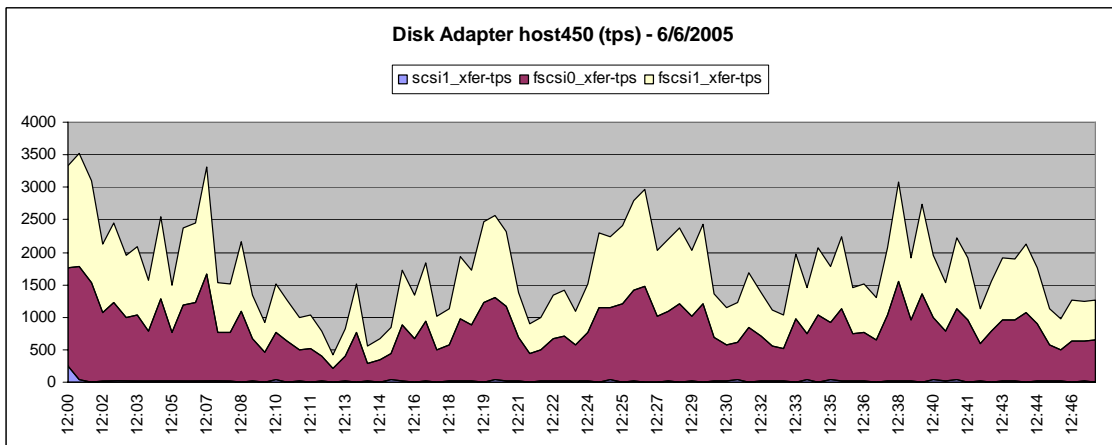
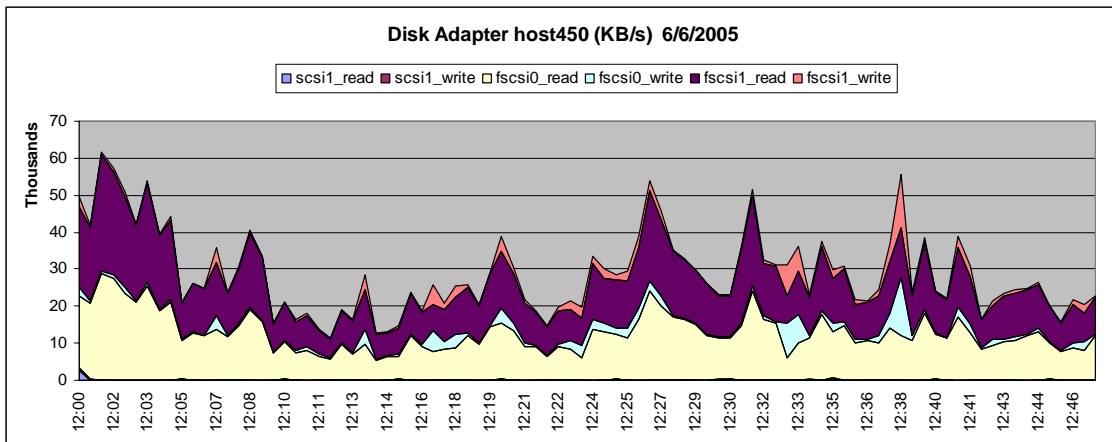
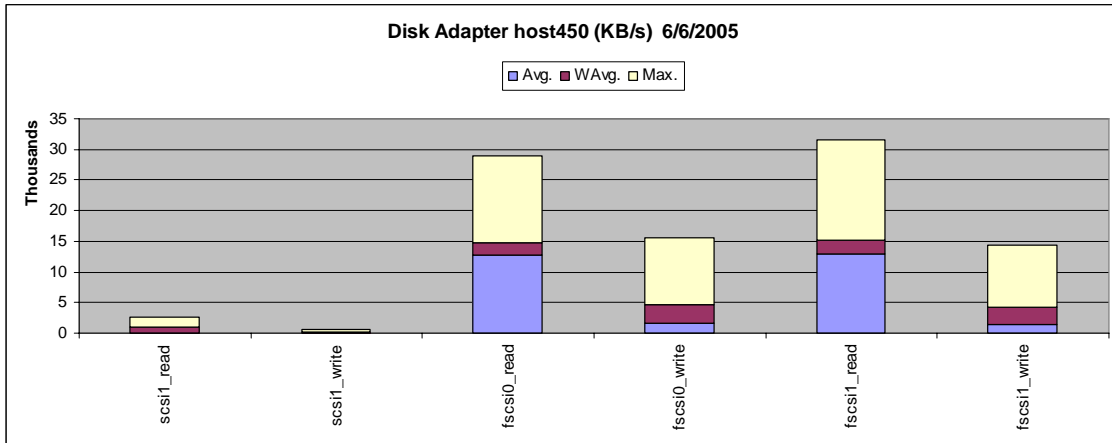


And also in the Write I/O statistics from the **EMCWRITE** worksheet:





Under heavy I/O conditions, individual I/O adapters can potentially be a capacity bottleneck. The **IOADAPT** worksheet provides information about adapter level I/O activity. This information should be checked to determine if the I/O activity is well-balanced across all of the available adapters. If not, the multi-path configuration might not be set up correctly. It is also a good idea to verify that the peak I/O throughput rates are well within the rated throughput capacity of the I/O adapters being used.



In some situations, I/O related bottlenecks can occur at the file system or device driver level levels due to an insufficient number of filesystem “bufstructs”. The [BBBP](#) worksheet provides ‘vmstat -v’ information at the beginning and again at the end (provided NMON completes normally, or is terminated with the “kill -USR2 pid” command) of the NMON run. The following example shows a fair number of “filesystem I/Os blocked with no fsbuf” events. If this value increases significantly over time, it is usually a good indication that the vmo “numfsbufs” parameter value (for JFS filesystems) needs to be increased. Similarly, if there are a significant (and increasing) number of “client filesystem I/Os blocked with no fsbuf” or “external pager filesystem I/Os blocked with no fsbuf” events, this may be an indication that the vmo “j2_nBufferPerPagerDevice” parameter (for JFS2 and other client based filesystems) value needs to be increased.

```
vmstat -v 8126464 memory pages
vmstat -v 7925882 lruable pages
vmstat -v 2840 free pages
vmstat -v 1 memory pools
vmstat -v 237680 pinned pages
vmstat -v 80.1 maxpin percentage
vmstat -v 15.0 minperm percentage
vmstat -v 30.0 maxperm percentage
vmstat -v 71.1 numperm percentage
vmstat -v 5641283 file pages
vmstat -v 0.0 compressed percentage
vmstat -v 5 compressed pages
vmstat -v 0.0 numclient percentage
vmstat -v 30.0 maxclient percentage
vmstat -v 10 client pages
vmstat -v 0 remote pageouts scheduled
vmstat -v 6816 pending disk I/Os blocked with no pbuf
vmstat -v 988112 paging space I/Os blocked with no psbuf
vmstat -v 161023 filesystem I/Os blocked with no fsbuf
vmstat -v 0 client filesystem I/Os blocked with no fsbuf
vmstat -v 0 external pager filesystem I/Os blocked with no fsbuf
```



8.2.4. Network

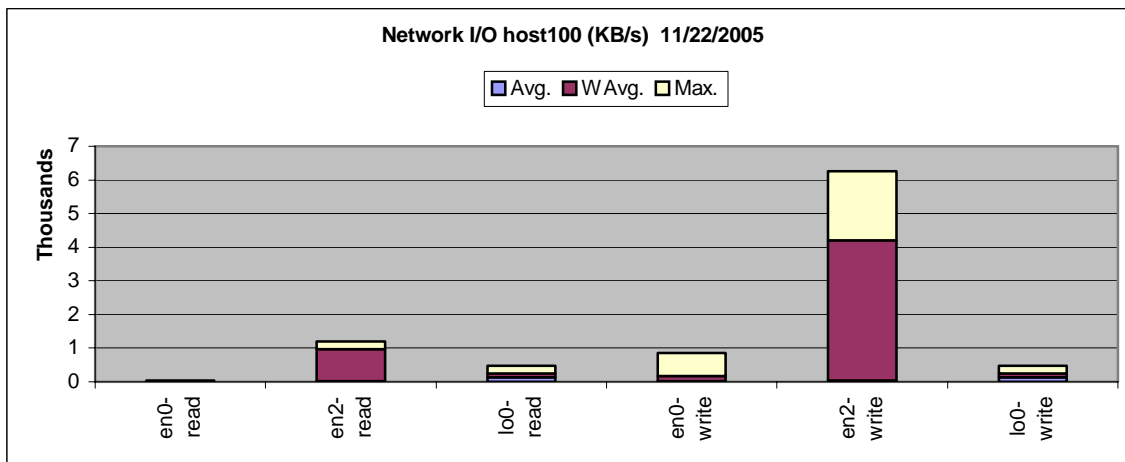
As long as network tuning best practices are followed, Oracle DB performance problems are rarely network related. It is always a good idea to check the BBBP worksheet to verify that the Network Options ('no') parameters are set appropriately. Common network settings for Oracle DB environments include the following:

- rfc1323=1
- tcp_sendspace>=262144
- tcp_recvspace>=262144
- sb_max=1048576

For Oracle Real Application Clusters (RAC) environments, the following parameters are also normally set:

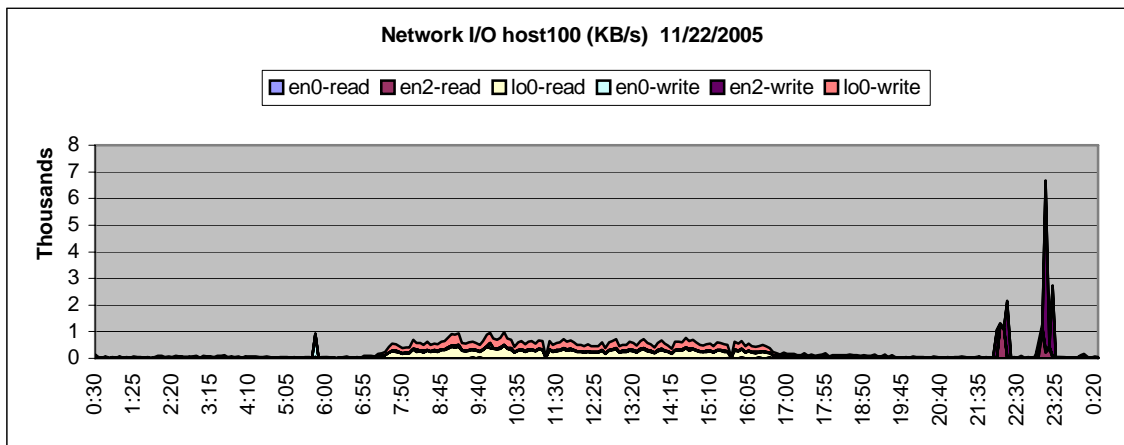
- $udp_sendspace = \max(65536, (db_block_size * db_file_multiblock_read_count) + 4096)$
- $udp_recvspace \geq 4 * udp_sendspace$

The NET worksheet provides information about total network throughput. The first graph provides average, weighted average and maximum throughput rates by adapter for read as well as write activity.





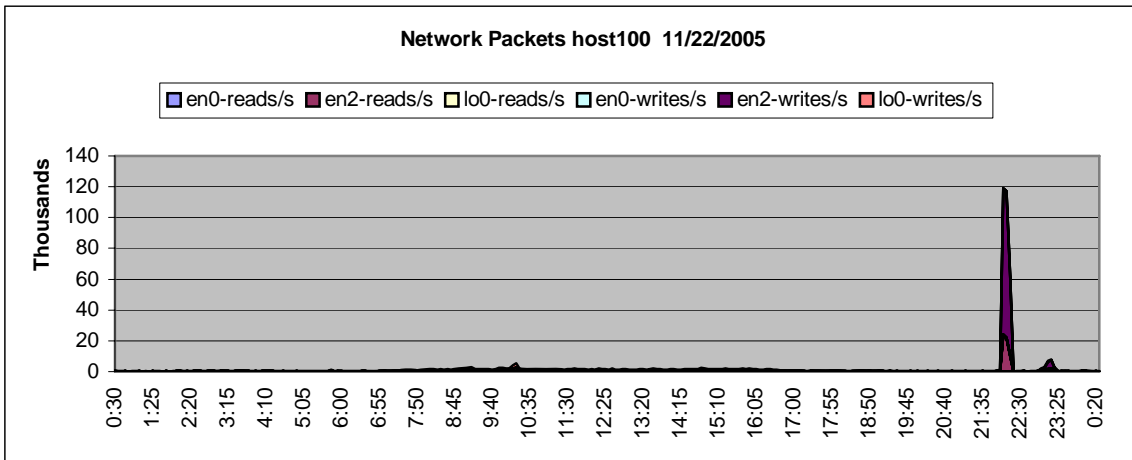
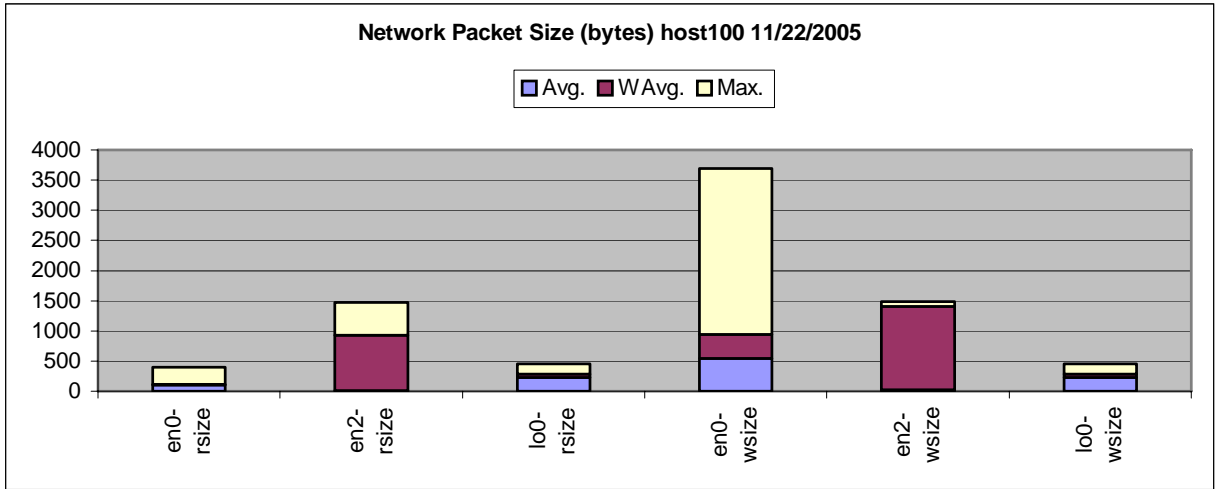
The second graph on the **NET** worksheet provides a system wide view of network throughput over the entire collection interval.



In this example, the system wide peak throughput is under 7 MB/second. This is well under the expected throughput capacity of even a single Gigabit Ethernet adapter (or even a 100 Mbps adapter), so network bandwidth (at least from a server perspective) is probably not an issue here. NMON has no knowledge of anything going on external to the AIX/IBM System p server environment, so this does not necessarily rule out potential issues within the network itself.



The **NETPACKET** worksheet provides additional information about average, weighted average and maximum packet sizes as well as the total number of packets sent/received per second.





9. Analyzing Oracle Statspack Data

9.1. Statspack Report Summary

At the beginning of each Statspack report, there is a summary that indicates the name of the Oracle DB instance, the Oracle version/patch level and the time period covered by the report. Ideally, for performance analysis purposes, a given Statspack report should not be for more than a 15-60 minute period. When statistics are averaged over longer reporting intervals, issues that occur over shorter time horizons tend to get diluted in the overall statistics. The Snap Id for the start and end of the report are also shown.

If the numbers are not consecutive, it means that multiple snapshots were taken during time period this Statspack report covers. If more detail is desired, a separate Statspack report can be generated for each combination of consecutive Begin Snap – End Snap values.

STATSPACK report for					
DB Name	DB Id	Instance	Inst Num	Release	Cluster
db100	1801685194	db100	1	9.2.0.5.0	NO
	Snap Id	Snap Time	Sessions	Curs/Sess	
Begin Snap:	18	02-Dec-04 17:24:48	685	123.4	
End Snap:	19	02-Dec-04 17:41:55	684	128.8	
Elapsed:		17.12 (mins)			

9.2. The “Load Profile” Section

Provides some high level statistics on Oracle DB activity and can be used to get a sense of the workload being run. For example, is there a lot of Redo log activity? What is the ratio of Physical Reads to Physical Writes?

Note that the Physical Reads and Physical Writes reported are the number DB blocks read or written. If Oracle is doing any multi-block read I/O operations, the number of physical read I/Os observed at the AIX or I/O subsystem level may be significantly lower than the number of blocks read. These are I/O rates as perceived by Oracle. If file system cache is being used, the physical I/O rates seen at the Operating System (NMON) level may be different, due to file system read cache hits, sequential read ahead operations, I/O coalesce or split operations, etc.

Load Profile ~~~~~	Per Second -----	Per Transaction -----
Redo size:	6,613,904.85	699,822.82
Logical reads:	178,971.31	18,937.10
Block changes:	21,231.97	2,246.57
Physical reads:	7,571.20	801.12
Physical writes:	2,246.41	237.69
User calls:	798.91	84.53
Parses:	410.75	43.46
Hard parses:	0.00	0.00
Sorts:	127.89	13.53
Logons:	0.00	0.00
Executes:	1,043.90	110.46
Transactions:	9.45	
% Blocks changed per Read:	11.86	Recursive Call %: 27.77
Rollback per transaction %:	0.00	Rows per Sort: 7.98



9.3. The “Instance Efficiency Percentages” Section

Provides high level information about Buffer Cache and Library Cache hit ratios as well as the percent of sorts that were done entirely within memory. Since workloads vary, there is no single good or bad cache hit rate for all applications. However, for a given environment, these hit ratios can be used as a general gauge to determine if the workload is changing over time or if cache related tuning is having the desired effect.

Instance Efficiency Percentages (Target 100%)			
Buffer Nowait %:	99.99	Redo NoWait %:	99.99
Buffer Hit %:	95.77	In-memory Sort %:	100.00
Library Hit %:	100.01	Soft Parse %:	100.00
Execute to Parse %:	60.65	Latch Hit %:	99.79
Parse CPU to Parse Elapsd %:	98.40	% Non-Parse CPU:	99.58
Shared Pool Statistics			
	Begin	End	
	-----	-----	
Memory Usage %:	11.50	11.63	
% SQL with executions>1:	92.70	92.71	
% Memory for SQL w/exec>1:	92.82	92.78	

9.4. The “Top 5 Timed Events” Section

Shows the top 5 Oracle wait events.

The following example is fairly typical of many OLTP applications. The predominant wait event is “db file sequential read”. This indicates single block random read activity, such as might occur when reading individual rows via a unique index. This workload would be considered “I/O bound”. The greatest opportunity for performance improvement for I/O bound workloads is in either avoiding the I/O entirely (through buffer cache tuning or application redesign) or in reducing the I/O response time through data layout or I/O subsystem related changes.

Top 5 Timed Events			
Event	Waits	Time (s)	% Total Ela Time
db file sequential read	7,751,940	92,849	82.03
CPU time		16,097	14.22
log file sync	11,086	1,923	1.70
rdbms ipc reply	2,152	947	.84
buffer busy waits	20,335	581	.51

The following example shows an environment with a significant amount of enqueue waits and relatively high CPU consumption. In this case, there is minimal I/O wait time reported. Areas of interest in this Statspack report would be the “Enqueue activity” section to determine what type of enqueue wait events is responsible for the bulk of the enqueue time and the “SQL ordered by Gets” section to determine if there are any SQL statements that might benefit from an appropriate index or a statement rewrite.

Top 5 Timed Events			
Event	Waits	Time (s)	% Total Ela Time
enqueue	3,283	822	67.61
CPU time		301	24.73
log file sync	27,937	57	4.72
direct path read (lob)	26,327	16	1.30
control file parallel write	2,016	4	.36

Whenever a new type of wait event shows up in the Top 5 Time Events report that you are not familiar with, it is always advisable to do some research on that wait event (such as doing a Metalink or google search) to understand what implications that wait event has on performance and what the possible causes (and solutions) are for that wait event. The following chart shows some of the wait events that might show up in the Top 5 Timed Events list and some investigative actions that might be appropriate based on those wait events:

Wait Event	Possible Action
buffer busy waits	<p>Indicates that another session is holding an incompatible lock on a database block or in the process of reading it into cache.</p> <p>-> Review the Top 10 Buffer Busy Waits per Segment section to determine which objects are experiencing the most contention. See ““WAITEVENT: "buffer busy waits" Reference Note”, Metalink Note # 34405.1 for suggestions on possible actions to take, depending on the block type.</p>
CPU time	<p>Indicates a “CPU bound” application.</p> <p>-> Review the “SQL ordered by Gets” section to determine if any of the listed SQL statements can be “tuned” to obtain a better access patch or be written more efficiently.</p>
db file sequential read	<p>Indicates significant single-block random read I/O activity (for example index lookups)</p> <p>-> Review the Buffer Pool Advisory section to determine if the Oracle Buffer Cache is adequately sized.</p> <p>-> Review the SQL ordered by Reads section to determine if any of the listed SQL statements can be “tuned” or rewritten to reduce the number of physical I/Os required.</p> <p>-> Review the Wait Events section to determine the average db file sequential read response time. If it appears high, review the Tablespace IO Stats and File IO Stats sections to see if particular tablespaces or files are much worse than others. Also review NMON or filemon data to determine if there is unbalanced I/O activity across devices, filesystem bufstruct bottlenecks, etc.</p>
db file scattered read	<p>Indicates significant multi-block sequential read I/O activity (for example tablespace scans)</p> <p>-> Review the SQL ordered by Reads section to determine if any of the listed SQL statements can be “tuned” or rewritten to reduce the number of physical I/Os required.</p> <p>-> Review the Wait Events section to determine the average db file scattered read response time. If it appears high, review the Tablespace IO Stats and File IO Stats sections to see if particular tablespaces or files are much worse than others. Also review NMON or filemon data to determine if there is unbalanced I/O activity across devices, filesystem bufstruct bottlenecks, etc.</p>

enqueue	<p>Indicates significant enqueue related activity.</p> <p>-> Review the Enqueue Activity section to determine what the predominant enqueue events are. 'TX' enqueues indicate there is row level lock contention. If these are seen, review the Top 10 Row Lock Waits per Segment section to determine which objects are experiencing the greatest amount of row lock contention.</p> <p>-> Review "FAQ about Detecting and Resolving Locking Conflicts", Metalink Note # 15476.1 for suggestions on how to deal with enqueue/lock contention issues.</p>
global cache cr request	<p>Indicates this is a RAC cluster and a lot of time is being spent retrieving a data block from global cache (on another node) to satisfy a Consistent Read (CR) request.</p> <p>-> Review the Cluster Statistics section to see what the average global cache get time is. A high average time can be an indication of contention, system load or network interconnect issues. Review NMON data to verify AIX Network Options (no) UDP_SENDSPEACE and UDP_RECVSPACE settings. Also look for potential Network throughput/capacity or CPU capacity issues. Otherwise, investigate the SQL responsible for the CR waits.</p>
latch free	<p>Indicates a significant amount of latch contention.</p> <p>-> Review "What are Latches and What Causes Latch Contention", Oracle Metalink note # 22908.1 for suggestions on how to analyze and resolve latch contention issues.</p>
log file sync	<p>Indicates that transactions are committing which still have redo log records in the log buffer cache.</p> <p>-> Review redo log write performance</p> <p>-> Review log buffer cache size to determine if it is too large, resulting in log buffer not getting full enough to trigger asynchronous redo log write activity.</p>
log buffer space	<p>Indicates the log buffer is filling up before redo log entries can be written to disk.</p> <p>-> Review redo log write performance</p> <p>-> Review log buffer cache size to determine if it is large enough to support the volume of redo log write activity</p>

9.5. The “Cluster Statistics” Section

When the Oracle instance is part of a RAC cluster, this section provides a high level statistics on Global Cache Service (GCS) and Global Enqueue Service (GES) events. Each Statspack report provides information for a single node only. In order to get a complete picture of overall RAC cluster performance, a separate Statspack report must be run for each of the instances (nodes) in the cluster.

```

Cluster Statistics for DB: db900 Instance: db900n1 Snaps: 2518 -2520

Global Cache Service - Workload Characteristics
-----
Ave global cache get time (ms):                7.4
Ave global cache convert time (ms):            0.1

Ave build time for CR block (ms):              0.0
Ave flush time for CR block (ms):              0.5
Ave send time for CR block (ms):              0.1
Ave time to process CR block request (ms):     0.6
Ave receive time for CR block (ms):            0.0

Ave pin time for current block (ms):          0.0
Ave flush time for current block (ms):         0.0
Ave send time for current block (ms):         0.0
Ave time to process current block request (ms): 0.1
Ave receive time for current block (ms):      16.1

Global cache hit ratio:                        1.6
Ratio of current block defers:                 0.0
% of messages sent for buffer gets:           1.3
% of remote buffer gets:                      0.3
Ratio of I/O for coherence:                   1.2
Ratio of local vs remote work:                4.0
Ratio of fusion vs physical writes:           0.3

Global Enqueue Service Statistics
-----
Ave global lock get time (ms):                0.0
Ave global lock convert time (ms):            0.1
Ratio of global lock gets vs global lock releases: 1.0

GCS and GES Messaging statistics
-----
Ave message sent queue time (ms):             0.0
Ave message sent queue time on ksxp (ms):     1.9
Ave message received queue time (ms):         0.0
Ave GCS message process time (ms):            0.0
Ave GES message process time (ms):            0.0
% of direct sent messages:                    49.7
% of indirect sent messages:                  50.1
% of flow controlled messages:                0.2

```



If Global Cache (gc) related events account for a high percentage of the total wait event activity (from [The “Top 5 Timed Events” Section](#)), the cluster statistics information should be reviewed in detail for activities with high average response times, for example “Ave global cache get time”, or “Ave receive time for current block”. Typically, these should be under 10 milliseconds (preferably < 2 milliseconds).

9.6. The “GES Statistics” Section

When the Oracle instance is part of a RAC cluster, this section provides Global Enqueue Service (GES) statistics. This information is for a single node only. In order to get a complete picture of overall RAC cluster performance, Statspack reports need to be run for each of the instances (nodes) in the cluster.

GES Statistics for DB: db900 Instance: db900n1 Snaps: 2518 -2520			
Statistic	Total	per Second	per Trans
...			
gcs compatible cr basts (global)	13	0.0	0.0
gcs compatible cr basts (local)	258,960	143.5	229.4
gcs cr basts to PIs	0	0.0	0.0
gcs cr serve without current lock	0	0.0	0.0
gcs error msgs	0	0.0	0.0
gcs flush pi msgs	173	0.1	0.2
gcs forward cr to pinged instance	0	0.0	0.0
gcs immediate (compatible) conver	48	0.0	0.0
gcs immediate (null) converts	238	0.1	0.2
gcs immediate cr (compatible) con	254,238	140.9	225.2
gcs immediate cr (null) converts	520,312	288.3	460.9
gcs msgs process time(ms)	33,597	18.6	29.8
gcs msgs received	1,049,383	581.4	929.5
...			
gcs side channel msgs actual	5,086	2.8	4.5
gcs side channel msgs logical	508,600	281.8	450.5
gcs write notification msgs	5	0.0	0.0
gcs write request msgs	34	0.0	0.0
gcs writes refused	0	0.0	0.0
ges msgs process time(ms)	94	0.1	0.1
ges msgs received	3,962	2.2	3.5
implicit batch messages received	119,596	66.3	105.9
implicit batch messages sent	115,905	64.2	102.7
lmd msg send time(ms)	53	0.0	0.0
lms(s) msg send time(ms)	1	0.0	0.0
messages flow controlled	2,440	1.4	2.2
messages received actual	773,734	428.7	685.3
messages received logical	1,053,374	583.6	933.0
messages sent directly	509,882	282.5	451.6
messages sent indirectly	514,567	285.1	455.8
msgs causing lmd to send msgs	1,851	1.0	1.6
msgs causing lms(s) to send msgs	5,368	3.0	4.8
msgs received queue time (ms)	17,711	9.8	15.7
msgs received queued	1,053,317	583.6	933.0
msgs sent queue time (ms)	14,533	8.1	12.9
msgs sent queue time on ksxp (ms)	1,462,576	810.3	1,295.5
msgs sent queued	514,586	285.1	455.8
msgs sent queued on ksxp	753,068	417.2	667.0
process batch messages received	306	0.2	0.3
process batch messages sent	265	0.1	0.2

9.7. The “Wait Events” Section

Provides additional details for the Oracle wait events, which are ranked in descending order by total wait time.

The average wait time (“Avg wait (ms)”) statistic is particularly useful when looking at I/O related wait times. In this particular example, the average wait time for “db file sequential read” was 12 milliseconds. This is the average single block read I/O response time as perceived by Oracle. Historically, read I/O response times below 20 milliseconds were considered good. Read I/O response times in the 10 – 20 millisecond range are typically reasonable for cache hostile workloads or I/O subsystems with limited read cache. However, many Enterprise class I/O subsystems sold today have relatively large read caches (many Gigabytes). When these types of I/O subsystems are used, it is not uncommon to see average wait times for “db file sequential read” in the 2 – 5 millisecond range for cache friendly workloads. If the I/O subsystem has write cache, write I/O response times should generally be below 2 milliseconds.

Write I/O response times significantly higher than this are typically a sign of a write I/O bottleneck. In addition to physical disk or adapter level throughput bottlenecks, poor write I/O performance can also be caused by file system inode contention or JFS2 CIO demotion events. If NMON does not show an obvious hdisk or file system (bufstructs) level bottleneck, more detailed AIX traces (for example filemon) may be required to determine the root cause.

```
Wait Events for DB: db100 Instance: db100 Snaps: 18 -19
-> s - second
-> cs - centisecond - 100th of a second
-> ms - millisecond - 1000th of a second
-> us - microsecond - 1000000th of a second
-> ordered by wait time desc, waits desc (idle events last)
```

Event	Waits	Timeouts	Total Wait Time (s)	Avg wait (ms)	Waits /txn
db file sequential read	7,751,940	0	92,849	12	798.7
log file sync	11,086	417	1,923	173	1.1
buffer busy waits	20,335	193	581	29	2.1
log file sequential read	6,156	0	257	42	0.6
db file parallel read	4,565	0	164	36	0.5
SQL*Net more data to client	305,568	0	126	0	31.5
log file parallel write	8,731	8,303	84	10	0.9
log file switch completion	631	0	35	55	0.1
latch free	7,572	7,013	29	4	0.8
async disk IO	12,255	0	26	2	1.3
db file parallel write	605	0	19	31	0.1
...					

9.8. The “Background Wait Events” Section

Provides statistics on wait events for Oracle background processes.

These wait events do not necessarily have a direct impact on application performance, so this section can often be ignored.

```
Background Wait Events for DB: db100 Instance: db100 Snaps: 18 -19
-> ordered by wait time desc, waits desc (idle events last)
```

Event	Waits	Timeouts	Total Wait Time (s)	wait (ms)	Avg Waits /txn
rdbms ipc reply	2,141	107	948	443	0.2
log file sequential read	6,156	0	257	42	0.6
log file parallel write	8,726	8,298	84	10	0.9
async disk IO	12,255	0	26	2	1.3
db file parallel write	605	0	19	31	0.1
control file parallel write	558	0	6	11	0.1
direct path read	3,848	0	5	1	0.4
direct path write	912	0	5	5	0.1
enqueue	25,290	2	3	0	2.6
...					



9.9. The “SQL ordered by Gets” Section

Shows the top SQL statements ranked by the total number of Logical Gets performed.

There tends to be a reasonably high correlation between Logical Get activity and Oracle CPU consumption. Therefore, when CPU resource usage is an issue, this section can provide useful information regarding those SQL statements that may be responsible for large amounts of CPU consumption and may be tuning candidates (for example index changes or SQL statement rewrite). For example, the first SQL statement shown consumed over 2,330 seconds (28.9% of the total) of CPU time over 258,000+ executions. The second SQL statement shown consumed 847 seconds of CPU time and averaged more than 10,000 Logical Gets per Execution.

```

SQL ordered by Gets for DB: db100 Instance: db100 Snaps: 18 -19
-> End Buffer Gets Threshold: 10000
-> Note that resources reported for PL/SQL includes the resources used
    by all SQL statements called within the PL/SQL code. As individual
    SQL statements are also reported, it is possible and valid for the
    Summed total % to exceed 100

Buffer Gets      Gets      CPU Elapsd
Executions per Exec %Total Time (s)  Time (s)  Hash Value
-----
53,073,448      258,331    205.4    28.9  2330.79    9647.61  521478685
INSERT INTO CAT_DOG_AST_PAW_ACVY (CAT_PAW_AMT, DOG_AST_PAW_ID
, PAW_TYP_CD, CAT_DOG_ST_ID, CAT_DOG_AST_PAW_ACVY_ID, CAT_PAW_
ACVY_PEN_TYP_CD, PAW_ACVY_ACTG_IMPC_IND, PAW_ACTG_TRMT_TYP_CD,
PAW_EAT_DSBT_DT, EAT_EAR_MTHD_CD, VER_NO, EAR_CYL_ST_ID, FNC
L_AST_ID) VALUES (:1, :2, :3, :4, :5, :6, :7, :8, :9, :10, :11,

31,753,730      3,143 10,103.0 17.3    847.95    2484.17  3252745759
UPDATE CAT_DOG_ST SET CAT_DOG_ST_EFF_DT = :1 , CAT_DOG_ST_PRPD_P
ER_CNT = :2 , CAT_DOG_PCSG_ST_CD = :3 , CAT_UPB_AMT = :4 , CAT_PR_
UPB_AMT = :5 , CAT_RMNG_TERM = :6 , CAT_AGE_NO = :7 , CAT_ACCT_PAY_
PEN_AMT = :8 , CAT_ACCT_RCVB_PEN_AMT = :9 , CAT_COMB_LTV_RTO_PCT =
:10 , CAT_ACCRD_INT_PEN_AMT = :11 , CAT_LPI_DT = :12 , CAT_TMLY_PM

19,469,486      23,827    817.1    10.6  1479.36    9273.88  2589488482
SELECT MAN.DOG_AST_ID, MAN.CAT_DOG_AST_PAW_ACVY_ID, MAN.EAR_CY
L_ST_ID, MAN.PAW_ACVY_ACTG_IMPC_IND, MAN.PAW_ACTG_TRMT_TYP_CD,
MAN.CAT_PAW_AMT, MAN.PAW_EAT_DSBT_DT, MAN.PAW_TYP_CD, MAN.FN
CL_AST_PAW_ID, MAN.CAT_DOG_ST_ID, MAN.CAT_PAW_ACVY_PEN_TYP_CD,
MAN.EAT_EAR_MTHD_CD, MAN.VER_NO FROM CAT_DOG_AST_PAW_ACVY WLO
...
    
```



The following example comes from an Oracle 11i E-Business Suite Environment. In this case, a single SQL statement was responsible for nearly 35% of the system Buffer Get activity. After opening a TAR regarding this particular SQL statement, Oracle Support was able to identify it as a known 11i issue. Once the appropriate fix was applied, the Buffer Get activity for this statement dropped by 80%, leading to a system wide reduction in Buffer Get activity of nearly 30% (along with a corresponding drop in system CPU consumption).

```

SQL ordered by Gets for DB: db300 Instance: db300 Snaps: 121 - 123
-> End Buffer Gets Threshold: #####
-> Note that resources reported for PL/SQL includes the resources used by
    all SQL statements called within the PL/SQL code. As individual SQL
    statements are also reported, it is possible and valid for the summed
    total % to exceed 100

          Gets          CPU  Elapsed
Buffer Gets  Executions per Exec %Total Time (s)  Time (s)  Hash Value
-----
 835,094,143      11,806 70,734.7   34.8  2919.97    4342.70 1101621696
Module: XXQP_OTBV_PLP_PRICING_WORKER
SELECT /*+ ORDERED USE_NL(qpq qplh qph)  index(qpq qp_qualifiers
_n4) index(qph qp_list_headers_b_n2) index(qplatq qp_preq_line_a
ttrs_tmp_n2) l_outer_qual_cur_mod */      qpq.list_header_id,
      qpq.list_line_id list_line_id,      qpq.qualifier_group
_cnt,      qpq.others_group_cnt,      qpq.header_qual_exists
    
```



9.10. The “SQL ordered by Reads” Section

Shows the top SQL statements ranked by the total number of Physical Reads performed.

For I/O bound workloads, this section can provide useful information regarding those SQL statements that may be responsible for large amounts of read I/O activity and may be tuning candidates (for example index changes or SQL statement rewrite). For example, the first SQL statement shown was responsible for more than 1 million physical reads and (for all executions) took 11,821 seconds (> 3 hours) of elapsed time to execute. The second statement was responsible for over 4.5 hours of elapsed time.

```
SQL ordered by Reads for DB: db100 Instance: db100 Snaps: 18 -19
-> End Disk Reads Threshold: 1000
```

Physical Reads	Executions	Reads per Exec	%Total	CPU Time (s)	Elapsd Time (s)	Hash Value
1,083,210	9,682	111.9	13.9	1336.78	11821.39	2153936339
SELECT MAN.DOG_AST_PAW_DEF_ID, MAN.PAW_ADV_TYP_CD, MAN.PAW_BAS_TYP_CD, MAN.DOG_AST_PAW_DEF_EFF_DT, MAN.DOG_AST_PAW_DEF_EXPT_DT, MAN.PAW_EAT_DT_TYP_CD, MAN.PAW_EAT_DAY_NO, MAN.PAW_EAT_FRQY_TYP_CD, MAN.PAW_EAT_TYP_CD, MAN.PAW_TYP_CD, MAN.FAC_F_FNM_ACQRD_PCT_IMPC_IND, MAN.DOG_AST_PAW_ACTG_IMPC_IND, MAN.F						
951,200	3,267	291.2	12.2	1554.49	15887.85	3844861954
SELECT MAN.CAT_DOG_ST_ID, MAN.CAT_ACCT_PAY_PEN_AMT, MAN.CAT_ACCT_RCVB_PEN_AMT, MAN.DOG_AST_ACCRL_PRIN_PEN_AMT, MAN.PRC DT_CLCTN_ST_ACVY_ID, MAN.CAT_COMB_LTV_RTO_PCT, MAN.CAT_DOG_ST_EFF_DT, MAN.CAT_FEES_PEN_AMT, MAN.CAT_AGE_NO, MAN.DOG_AST_ID, MAN.CAT_FST_YR_PC_SET_ASIDE_AMT, MAN.LAST_UPD_DT, MAN.CAT_ACCRD_INT_PEN_AMT, MAN.						
744,224	3,244	229.4	9.6	1036.12	9067.36	4171281628
SELECT MAN.DOG_AST_ID, MAN.TRAN_TYP_CD, MAN.ACP_CHG_TYP_DT, WLO.FNM_CAT_ID, MAN.CAT_DOG_ACVY_ARCHV_DT, MAN.CAT_LPI_AT_SIR_SETUP_DT, MAN.CAT_EAT_TYP_CD, MAN.CAT_PRDC_LBL_TYP_CD, MAN.CAT_STAT_CD, WLO.CAT_FNM_ACQRD_PCT, MAN.CAT_VER_NO, MAN.LPC_EAR_CYL_ID, MAN.LPC_PRC DT_CLCTN_ID, WL1.CAT_EAR_RT_FEAT_ID, WL1.CAT_INT_ACCRL_FRQY_						
...						

9.11. The “Instance Activity Stats” Section

Provides a lot of additional statistics regarding Oracle instance activity.

Depending on the particular issue being analyzed, there might be useful information here. For example, if there appear to be redo log I/O performance issues, you might want to look at the redo log related statistics. Or, if there are potential network response time issues, you might want to look at the SQL*Net related statistics.

Instance Activity Stats for DB: db100 Instance: db100 Snaps: 18 -19			
Statistic	Total	per Second	per Trans
CPU used by this session	1,609,668	1,567.4	165.8
CPU used when call started	1,609,312	1,567.0	165.8
CR blocks created	51,270	49.9	5.3
Cached Commit SCN referenced	1,438,332	1,400.5	148.2
Commit SCN cached	275	0.3	0.0
DBWR buffers scanned	7,657,735	7,456.4	789.0
DBWR checkpoint buffers written	1,652,662	1,609.2	170.3
DBWR checkpoints	3	0.0	0.0
DBWR free buffers found	7,052,384	6,867.0	726.6
DBWR lru scans	477	0.5	0.1
DBWR make free requests	2,236	2.2	0.2
DBWR revisited being-written buff	17	0.0	0.0
DBWR summed scan depth	7,657,735	7,456.4	789.0
DBWR transaction table writes	1,037	1.0	0.1
DBWR undo block writes	326,236	317.7	33.6
...			

9.12. The “Tablespace IO Stats” Section

Provides a summary of Read I/O and Write I/O activity by Tablespace, ordered by total (Read + Write) I/Os.

The I/Os reported here are the actual I/O requests made to the Operating System. (Unlike “Load Profile” section which reported I/O activity in DB blocks, not actual I/O requests.) The “Av Blks/Rd” statistics is the average number of DB blocks read per I/O request. If there is any multi-block read activity (“db file scattered read”) for a given tablespace, the reported value will be greater than 1.0. It is not unusual for multi-block reads (“db file scattered read”) to have higher average responses times than single block reads (“db file sequential read”), because more data is being returned in a single I/O request. Therefore, it is not unusual for tablespaces with a high “Av Blks/Rd” value to show somewhat higher average read response times than tablespaces with a low “Av Blks/Rd” value. On the other hand, if the filesystem or the I/O subsystem is able to detect sequential read (“db file scattered read”) activity, these tablespaces may show extremely good read I/O response times because the blocks have been prefetched into filesystem or I/O subsystem cache prior to the read I/O request from Oracle. If any individual tablespaces show particularly high average Read I/O response times, there may be data file placement or other I/O subsystem related performance issues.

This section also reports on database Buffer Busy Waits. A high number of buffer busy waits can be the result of I/O performance issues, or might be due to application level contention issues.

Since the tablespaces are listed in descending order of total I/O activity, the ones near the top of the list deserve much more attention than those near the bottom of the list. This is because those tablespaces with a lot of I/O activity will generally contribute a much large percentage of total I/O wait time than a tablespace with relatively little I/O activity.

It is not uncommon to see rather high Average Read I/O response times reported for infrequently accessed tablespaces/data files. Normally, tablespaces with infrequent access do not account for a large percentage of total Read I/O response time. Unless they appear near the top of the list, it is not generally worthwhile to investigate these.

```

Tablespace IO Stats for DB: db100 Instance: db100 Snaps: 18 -19
->ordered by IOs (Reads + Writes) desc

Tablespace
-----
          Av      Av      Av          Av      Buffer  Av Buf
          Reads Reads/s Rd(ms) Blks/Rd      Writes Writes/s      Waits  Wt(ms)
-----
CAT_DATA_CLUSTER_02
802,868      782    14.2      1.0    666,416      649      2,448      0.3
CAT_PART_DATA02_P11
359,653      350    22.9      1.0    253,422      247         64      5.5
CAT_PART_IDX_P08
399,569      389    10.6      1.0     40,927       40         8      5.0
UNDOTBS

```



85	0	20.8	1.0	327,109	319	1,157	508.6
CAT_PART_IDX_P09							
218,593	213	12.5	1.0	81,721	80	6	3.3
...							



9.13. The “File IO Stats” Section

Provides a summary of Read I/O and Write I/O activity at the individual file level (ordered by tablespace name and file name).

If one or more of the tablespaces near the top of the “Tablespace IO Stats” list shows high average Read I/O response times, it is a good idea to check the individual file statistics for those particular tablespace(s) to see if response times are consistent, or whether some individual files show significantly higher average read I/O response times than others.

As mentioned earlier, it is not uncommon to see rather high Average Read I/O response times reported for infrequently accessed data files. Unless these files account for a large percentage of total Read I/O response time, so it is not generally worthwhile to further investigate these.

```
File IO Stats for DB: db100 Instance: db100 Snaps: 18 -19
->ordered by Tablespace, File
```

Tablespace	Filename	Av Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
CAT_DATA_CLUSTER	/db/db008/oracle/CAT/cat_data_cluster.df02.dbf	40,160	39	18.0	1.0	24,541	24	4	2.5
	/db/db008/oracle/CAT/cat_data_cluster.df16.dbf	21,825	21	16.7	1.0	23,656	23	34	0.3
	/db/db009/oracle/CAT/cat_data_cluster.df07.dbf	39,678	39	13.9	1.0	25,476	25	17	0.0
	/db/db009/oracle/CAT/cat_data_cluster.df03.dbf	35,453	35	13.7	1.0	19,882	19	0	
	/db/db009/oracle/CAT/cat_data_cluster.df17.dbf	24,804	24	13.9	1.0	27,537	27	73	0.3
	/db/db010/oracle/CAT/cat_data_cluster.df04.dbf	36,594	36	13.5	1.0	21,484	21	1	0.0
	/db/db010/oracle/CAT/cat_data_cluster.df18.dbf	24,584	24	12.8	1.0	24,067	23	178	0.7
	/db/db011/oracle/CAT/cat_data_cluster.df05.dbf	32,462	32	13.5	1.0	18,653	18	0	
	/db/db011/oracle/CAT/cat_data_cluster.df19.dbf	27,691	27	13.6	1.0	32,154	31	229	0.2
	/db/db012/oracle/CAT/cat_data_cluster.df06.dbf	33,827	33	13.2	1.0	19,772	19	0	
...									

9.14. The “Buffer Pool Advisory”

Provides predictions on what impact changes in the Oracle Buffer Cache size would have on physical read I/O activity.

Below is an example of an environment with a relatively large current Buffer Cache allocation of 30 GB. According to the advisory, doubling the Buffer cache size to 60 GB could have reduced the Physical Reads required for this workload by about 15% (from 12,338,816 to 10,525,967). In this case, a 15% reduction in Oracle read I/O activity does not necessarily justify an investment in an additional 30 GB of physical memory. There is also likely to be some additional CPU overhead associated with managing the larger Buffer Cache size or other issues (for example block contention) which partially (or wholly) offset any Read I/O related performance gains.

```
Buffer Pool Advisory for DB: db100 Instance: db100 End Snap: 19
-> Only rows with estimated physical reads >0 are displayed
-> ordered by Block Size, Buffers For Estimate
```

P	Size for Estimate (M)	Size Factr	Buffers for Estimate	Est Physical Read Factor	Estimated Physical Reads
D	3,072	.1	381,120	1.53	18,826,375
D	6,144	.2	762,240	1.33	16,358,840
D	9,216	.3	1,143,360	1.21	14,958,222
D	12,288	.4	1,524,480	1.16	14,349,390
D	15,360	.5	1,905,600	1.13	13,929,032
D	18,432	.6	2,286,720	1.10	13,562,132
D	21,504	.7	2,667,840	1.07	13,223,857
D	24,576	.8	3,048,960	1.05	12,907,536
D	27,648	.9	3,430,080	1.02	12,616,480
D	30,720	1.0	3,811,200	1.00	12,338,816
D	33,792	1.1	4,192,320	0.98	12,082,370
D	36,864	1.2	4,573,440	0.96	11,842,298
D	39,936	1.3	4,954,560	0.94	11,621,420
D	43,008	1.4	5,335,680	0.93	11,418,821
D	46,080	1.5	5,716,800	0.91	11,232,627
D	49,152	1.6	6,097,920	0.90	11,059,913
D	52,224	1.7	6,479,040	0.88	10,906,528
D	55,296	1.8	6,860,160	0.87	10,763,880
D	58,368	1.9	7,241,280	0.86	10,638,895
D	61,440	2.0	7,622,400	0.85	10,525,967



The following is an example where the current Buffer Cache allocation appears to be much smaller than optimal. According to the Buffer Pool Advisory, it is likely that if the Buffer Cache is expanded from 512 MB to 960 MB (a 90% increase), the number of Physical Reads that need to be performed will drop by nearly 60%.

Given that more than 40% of the wait time reported in the [The “Top 5 Timed Events” Section](#) (from the same Statspack report) is due to read I/O activity, It is extremely likely that increasing the Buffer Cache size will have a significant positive effect on application performance.

However, remember that you should only increase the amount of Oracle SGA memory used if there is sufficient physical memory on the server to support the increase without causing system paging.

```
Buffer Pool Advisory for DB: db200 Instance: db200 End Snap: 45
-> Only rows with estimated physical reads >0 are displayed
-> ordered by Block Size, Buffers For Estimate
```

P	Size for Estimate (M)	Size Factr	Buffers for Estimate	Est Physical Read Factor	Estimated Physical Reads
D	48	.1	5,955	52.24	55,541,493,507
D	96	.2	11,910	48.78	51,856,997,429
D	144	.3	17,865	46.45	49,383,022,423
D	192	.4	23,820	45.26	48,114,613,024
D	240	.5	29,775	41.58	44,205,675,187
D	288	.6	35,730	5.27	5,602,647,833
D	336	.7	41,685	3.22	3,419,333,556
D	384	.8	47,640	2.15	2,281,656,878
D	432	.8	53,595	1.46	1,551,889,061
D	480	.9	59,550	1.15	1,227,569,750
D	512	1.0	63,520	1.00	1,063,124,041
D	528	1.0	65,505	0.94	997,803,270
D	576	1.1	71,460	0.80	855,214,608
D	624	1.2	77,415	0.71	754,815,162
D	672	1.3	83,370	0.64	675,830,967
D	720	1.4	89,325	0.58	616,785,509
D	768	1.5	95,280	0.53	566,820,285
D	816	1.6	101,235	0.50	528,862,634
D	864	1.7	107,190	0.47	496,928,919
D	912	1.8	113,145	0.44	469,598,544
D	960	1.9	119,100	0.42	444,185,471



9.15. The “PGA Memory Advisory” Section

(Oracle 9i and above) provides predictions on what impact changes in the Oracle `pga_aggregate_target` value would have on PGA memory efficiency.

Below is an example of an environment where the `pga_aggregate_target` is 4 GB (4,096 MB). Per the advisory, the minimum recommended target value is 2 GB (2,048 MB), since this is the smallest value shown where the “Estd PGA Overalloc Count” is zero. In addition, the advisory predicts that the PGA Cache Hit % could be improved from 98% to 100% if the `pga_aggregate_target` were increased from 4 GB to 4.8 GB (4,915 MB).

Since the `pga_aggregate_target` value is not a hard limit, it is often difficult to predict actual PGA memory usage based on the `pga_aggregate_target` setting. It is advisable to periodically monitor actual PGA memory usage (especially after changing the `pga_aggregate_target` value) as well as overall physical memory usage to ensure that the Oracle virtual memory requirements can be satisfied in physical memory – without inducing system paging.

```
PGA Memory Advisory for DB: db100 Instance: db100 End Snap: 19
-> When using Auto Memory Mgmt, minimally choose a pga_aggregate_target
value where Estd PGA Overalloc Count is 0
```

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overalloc Count
512	0.1	23,988.7	3,155.8	88.0	38
1,024	0.3	23,988.7	3,011.5	89.0	3
2,048	0.5	23,988.7	1,708.5	93.0	0
3,072	0.8	23,988.7	410.2	98.0	0
4,096	1.0	23,988.7	410.2	98.0	0
4,915	1.2	23,988.7	0.0	100.0	0
5,734	1.4	23,988.7	0.0	100.0	0
6,554	1.6	23,988.7	0.0	100.0	0
7,373	1.8	23,988.7	0.0	100.0	0
8,192	2.0	23,988.7	0.0	100.0	0
12,288	3.0	23,988.7	0.0	100.0	0
16,384	4.0	23,988.7	0.0	100.0	0
24,576	6.0	23,988.7	0.0	100.0	0
32,768	8.0	23,988.7	0.0	100.0	0



9.16. The “Enqueue Activity” Section

Provides a breakdown of system enqueue activity.

If “enqueue” wait events show up in the [Top 5 Timed Events](#) list, it is important to understand what the primary cause(s) of enqueue waits. In the following example, the largest category of Enqueue wait time is associated with TX enqueues. Generally, this has to do with row level latch contention within the application.

```

Enqueue activity for DB: db100 Instance: db100 Snaps: 18 -19
-> Enqueue stats gathered prior to 9i should not be compared with 9i
    data
-> ordered by Wait Time desc, Waits desc
    
```

Eq	Requests	Succ Gets	Failed Gets	Waits	Avg Wt Time (ms)	Wait Time (s)
TX	32,717	32,717	0	3,105	271.08	842
CF	1,221	1,223	0	6	542.00	3
HW	7,614	7,614	0	77	10.40	1
FB	15,615	15,615	0	38	1.03	0
SQ	1,296	1,296	0	1	6.00	0

9.17. The “Top 10 Buffer Busy Waits per Segment” Section

Provides details on the database objects having the highest number of buffer busy waits. In situations where buffer busy waits account for a large percentage of the total wait time shown in the “Top 5 Time Events” section, these are some of the objects where further analysis might be warranted.

Below is an example from an Oracle 11i E-Business Suite environment.

```

Top 10 Buf. Busy Waits per Segment for DB: db200 Instance: db200 124 - 126
-> End Segment Buffer Busy Waits Threshold: 100
    
```

Owner	Tablespace	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	%Total
APPS	AOL_DATA	ECX_OUTQUEUE		TABLE	474,164	24.80
ONT	ONT_DATA	OE_ORDER_LINES_ALL		TABLE	358,612	18.76
SYSTEM	SYSTEM	AQ\$_QUEUE_TABLES		TABLE	196,059	10.26
APPLSYS	AOL_DATA	FND_CONCURRENT_REQUE		TABLE	112,182	5.87
AR	AR_DATA	AR_PAYMENT_SCHEDULES		TABLE	105,189	5.50
APPLSYS	WF_IDX	WF_ITEM_ACTIVITY_STA		INDEX	55,651	2.91
APPLSYS	WF_DATA	WF_ITEM_ACTIVITY_STA	YS_SUBP530	TABLE	55,284	2.89
APPLSYS	WF_DATA	WF_ITEM_ACTIVITY_STA	YS_SUBP529	TABLE	54,666	2.86
SYS	SYSTEM	AQ\$_QUEUE_TABLE_AFFI		TABLE	52,129	2.73
APPLSYS	WF_DATA	WF_ITEM_ACTIVITY_STA	YS_SUBP531	TABLE	48,351	2.53

9.18. The “Top 10 Row Lock Waits per Segment” Section

Provides details on the database objects having the highest number of buffer busy waits.

In situations where there are significant ‘TX’ enqueues, these are the objects likely to warrant further analysis.

Below is an example from an Oracle 11i E-Business Suite environment.

```

Top 10 Row Lock Waits per Segment for DB: db200 Instance: db200
Snaps: 124 - 126
-> End Segment Row Lock Waits Threshold:      100

```

Owner	Tablespace	Object Name	Type	Waits	%Total
APPLSYS	AOL_DATA	FND_CONCURRENT_REQUE	TABLE	7,906	32.41
SYS	SYSTEM	AQ\$_QUEUE_TABLE_AFFI	TABLE	6,983	28.63
SYSTEM	SYSTEM	AQ\$_QUEUE_TABLES	TABLE	3,450	14.14
ASO	ASO_DATA	SYS_IOT_TOP_365627	INDEX	602	2.47
APPLSYS	AOL_DATA	FND_CONC_PROG_ONSITE	TABLE	565	2.32
APPLSYS	AOL_DATA	SYS_IOT_TOP_1105301	INDEX	557	2.28
APPLSYS	AOL_DATA	SYS_IOT_TOP_1105299	INDEX	484	1.98
WSH	WSH_DATA	WSH_DELIVERY_DETAILS	TABLE	406	1.66
JTF	JTF_DATA	JTF_PREFAB_CACHE_STA	TABLE	351	1.44



9.19. The “Shared Pool Advisory” Section

Provides predictions on what impact changes in the Oracle Shared Pool size would have on SGA memory efficiency.

In the following example, the Shared Pool appears to be somewhat oversized. The advisory is predicting the same performance when using a Shared Pool size of 2,448 MB as when using the current Shared Pool size of 4,112 MB. This memory might potentially be put to more efficient use elsewhere, such as the Oracle Buffer Cache.

```
Shared Pool Advisory for DB: VOSS07 Instance: VOSS07 End Snap: 19
-> Note there is often a 1:Many correlation between a single logical object
    in the Library Cache, and the physical number of memory objects associated
    with it. Therefore comparing the number of Lib Cache objects (for example
in
    v$sqlibrarycache), with the number of Lib Cache Memory Objects is invalid
```

Shared Pool Size for Estim (M)	SP Size Factr	Estd Lib Cache Size (M)	Estd Lib Cache Mem Obj	Estd Lib Cache Time Saved (s)	Estd LC Time Saved Factr	Estd Lib Cache Mem Obj Hits
2,448	.6	58	5,588	4,267	1.0	615,417
2,864	.7	58	5,588	4,267	1.0	615,417
3,280	.8	58	5,588	4,267	1.0	615,417
3,696	.9	58	5,588	4,267	1.0	615,417
4,112	1.0	58	5,588	4,267	1.0	615,417
4,528	1.1	58	5,588	4,267	1.0	615,417
4,944	1.2	58	5,588	4,267	1.0	615,417
5,360	1.3	58	5,588	4,267	1.0	615,417
5,776	1.4	58	5,588	4,267	1.0	615,417
6,192	1.5	58	5,588	4,267	1.0	615,417
6,608	1.6	58	5,588	4,267	1.0	615,417
7,024	1.7	58	5,588	4,267	1.0	615,417
7,440	1.8	58	5,588	4,267	1.0	615,417
7,856	1.9	58	5,588	4,267	1.0	615,417
8,272	2.0	58	5,588	4,267	1.0	615,417

9.20. The “SGA Memory Summary” Section

Provides a high level breakdown of how SGA memory is allocated.

Below is an example from a large Oracle DB environment, which has fairly significant amounts of Oracle Buffer Cache and SGA memory allocated.

SGA Memory Summary for DB: db100 Instance: db100 Snaps: 18 -19	
SGA regions	Size in Bytes
-----	-----
Database Buffers	38,654,705,664
Fixed Size	796,872
Redo Buffers	33,570,816
Variable Size	7,046,430,720
-----	-----
sum	45,735,504,072

9.21. The “SGA breakdown difference” Section

Provides component level detail of SGA memory allocation at the beginning and the end of the Statspack reporting interval. This can be useful to determine if any particular SGA components are growing in size. Components that consistently grow over time might have fragmentation or memory leak issues.

This section can also provide useful information concerning how optimal the current SGA related settings are. In this example, there appear to be significant amounts of free memory in the large pool and in the shared pool. This might mean that these areas are currently overallocated and the physical memory being used for the SGA might be better used elsewhere, for example more Buffer Cache, more PGA, etc...

SGA breakdown difference for DB: db100 Instance: db100 Snaps: 18 -19				
Pool	Name	Begin value	End value	% Diff
java	free memory	167,772,160	167,772,160	0.00
large	PX msg pool	117,964,800	117,964,800	0.00
large	free memory	2,029,518,848	2,029,518,848	0.00
shared	1M buffer	2,098,176	2,098,176	0.00
shared	BRANCH TABLE SEGMENTE	37,152	37,152	0.00
...				
shared	errors	29,936	29,936	0.00
shared	event statistics per se	49,582,680	49,582,680	0.00
shared	fixed allocation callba	3,032	3,032	0.00
shared	free memory	4,187,155,456	4,180,746,616	-0.15
shared	joxs heap init	4,240	4,240	0.00
shared	kglsim hash table bkts	4,194,304	4,194,304	0.00
...				
shared	ktlbk state objects	2,596,920	2,596,920	0.00
shared	library cache	45,180,024	48,042,912	6.34
shared	log_buffer	1,048,576	1,048,576	0.00
...				
	buffer_cache	38,654,705,664	38,654,705,664	0.00
	fixed_sga	796,872	796,872	0.00
	log_buffer	33,554,432	33,554,432	0.00

9.22. The “init.ora Parameters” Section

Provides details on the starting and ending Oracle initialization (init.ora) parameter settings. This is often a useful place to check on I/O related settings, for example db_file_multiblock_read count.

In general the use of, undocumented “hidden parameters” -- those that begin with the underscore (“_”) character -- is discouraged unless explicitly recommended by Oracle support.

When running vendor provided application packages, it is advisable to verify that init.ora settings are consistent with settings recommended by the application provider. For example, Oracle 11i E-Business Suite customers should refer to the “Database Initialization Parameters and Configuration for Oracle Applications 11i” note available in Metalink (Id # 216205.1).

```
init.ora Parameters for DB: db100 Instance: db100 Snaps: 18 -19
                                     End value
Parameter Name                       Begin value                       (if different)
-----
_dynamic_rls_policies                 FALSE
compatible                            9.2.0
control_files                         /db/db001/oracle/CAT/ctrl11CAT
core_dump_dest                        /export/oracle/dbsdumps/CAT/cd
cursor_sharing                        similar
cursor_space_for_time                TRUE
db_block_checksum                     FALSE
db_block_size                         8192
db_cache_advice                       ON
db_cache_size                         32212254720
db_file_multiblock_read_count        64
db_files                              500
db_keep_cache_size                   2147483648
db_name                               db100
fast_start_mttr_target                60
ifile                                 /export/oracle/admin/CAT/pfile
java_pool_size                        167772160
job_queue_processes                  4
large_pool_size                      2147483648
log_archive_dest                      /export/oracle/archive/CAT/log
log_archive_format                    arc_%t_%s.log
log_archive_max_processes             5
log_archive_start                     TRUE
log_buffer                            16777216
log_checkpoint_interval               2147483647
log_checkpoints_to_alert              TRUE
log_parallelism                       2
max_dump_file_size                   2097152
max_enabled_roles                     100
open_cursors                          1000
optimizer_mode                        CHOOSE
os_authent_prefix                     TRUE
pga_aggregate_target                  4294967296
processes                             4000
query_rewrite_integrity               stale_tolerated
recovery_parallelism                  8
session_cached_cursors                1100
shared_pool_reserved_size             213909504
shared_pool_size                      4294967296
timed_statistics                      TRUE
undo_management                       AUTO
undo_retention                        30
undo_tablespace                       undotbs
user_dump_dest                        /export/oracle/dbsdumps/CAT/ud
```

utl_file_dir	*
workarea_size_policy	auto

10. AWR Report Examples

In order to show similarities and differences between the Oracle Statspack and AWR reports, following are samples of selected sections from an AWR report.

10.1. The “Report Summary” Section

DB Name	DB Id	Instance	Inst num	Release	Cluster	Host
b321	1954197547	db321	1	10.1.0.4.0	YES	host321
	Snap Id	Snap Time	Sessions	Cursors/Session		
Begin Snap:	1117	29-Nov-05 13:00:12	281	52.9		
End Snap:	1118	29-Nov-05 14:00:17	291	54.3		
Elapsed:		60.08 (mins)				
DB Time:		354.74 (mins)				

10.2. The “Load Profile” Section

	Per Second	Per Transaction	
Redo size:	4,875.05	27,720.16	
Logical reads:	366,571.95	2,084,374.39	
Block changes:	1,077.95	6,129.38	
Physical reads:	120.76	686.64	
Physical writes:	8.85	50.33	
User calls:	875.84	4,980.13	
Parses:	146.83	834.89	
Hard parses:	0.12	0.70	
Sorts:	265.21	1,508.04	
Logons:	3.27	18.58	
Executes:	189.66	1,078.42	
Transactions:	0.18		
% Blocks changed per Read:	0.29	Recursive Call %:	16.07
Rollback per transaction %:	9.46	Rows per Sort:	547.81

10.3. The “Instance Efficiency Percentages” Section

Buffer Nowait %:	100.00	Redo NoWait %:	100.00
Buffer Hit %:	99.97	In-memory Sort %:	100.00
Library Hit %:	99.93	Soft Parse %:	99.92
Execute to Parse %:	22.58	Latch Hit %:	99.41
Parse CPU to Parse Elapsed %:	41.01	% Non-Parse CPU:	99.82

10.4. The “Top 5 Timed Events” Section

Event	Waits	Time(s)	Percent Total DB Time	Wait Class
CPU time		10,788	50.68	
enq: US - contention	921,839	964	4.53	Other
db file sequential read	285,394	731	3.43	User I/O
DFS lock handle	838,368	502	2.36	Other
gc current block 2-way	57,683	118	.55	Cluster

10.5. The “Cluster Statistics” Section

Global Cache Load Profile

	Per Second	Per Transaction
Global Cache blocks received:	45.39	258.11
Global Cache blocks served:	9.28	52.78
GCS/GES messages received:	1,601.07	9,103.90
GCS/GES messages sent:	1,392.84	7,919.88
DBWR Fusion writes:	0.28	1.57

Global Cache Efficiency Percentages (Target local+remote 100%)

Buffer access - local cache %:	99.95
Buffer access - remote cache %:	0.01
Buffer access - disk %:	0.03

Global Cache and Enqueue Services - Workload Characteristics

Avg global enqueue get time (ms):	1.3
Avg global cache cr block receive time (ms):	2.7
Avg global cache current block receive time (ms):	2.5
Avg global cache cr block build time (ms):	0.0
Avg global cache cr block send time (ms):	0.0
Global cache log flushes for cr blocks served %:	19.6
Avg global cache cr block flush time (ms):	1.3
Avg global cache current block pin time (ms):	0.1
Avg global cache current block send time (ms):	0.0
Global cache log flushes for current blocks served %:	0.1
Avg global cache current block flush time (ms):	1.3

Global Cache and Enqueue Services - Messaging Statistics

Avg message sent queue time (ms):	0.0
Avg message sent queue time on kxsp (ms):	0.9
Avg message received queue time (ms):	0.0
Avg GCS message process time (ms):	0.0
Avg GES message process time (ms):	0.0
% of direct sent messages:	70.36
% of indirect sent messages:	24.53
% of flow controlled messages:	5.11

10.6. The “GES Statistics” Section

Statistic	Total	per Second	per Trans
acks for commit broadcast(actual)	45,943	12.74	72.47
acks for commit broadcast(logical)	47,958	13.30	75.64
broadcast msgs on commit(actual)	8,057	2.23	12.71
broadcast msgs on commit(logical)	8,061	2.24	12.71
broadcast msgs on commit(wasted)	259	0.07	0.41
dynamically allocated gcs resources	0	0.00	0.00
dynamically allocated gcs shadows	0	0.00	0.00
false posts waiting for scn acks	6	0.00	0.01
flow control messages received	27	0.01	0.04
flow control messages sent	0	0.00	0.00
...			

10.7. The “Wait Events” Section

Event	Waits	Timeouts	Total Wait Time (s)	Avg wait (ms)	Waits /txn
enq: US – contention	921,839	0	964	1	1,454.00
db file sequential read	285,394	0	731	3	450.15
DFS lock handle	838,368	3	502	1	1,322.35
gc current block 2-way	57,683	6	118	2	90.98
gc cr block 2-way	46,864	39	102	2	73.92
gc cr grant 2-way	128,644	36	101	1	202.91
gc cr block 3-way	22,177	8	70	3	34.98
gc cr block busy	21,529	2	68	3	33.96
latch: cache buffers chains	3,058	3,055	53	17	4.82
db file parallel read	3,663	0	30	8	5.78
...					



10.8. The “Background Wait Events” Section

Event	Waits	Timeouts	Total Wait Time (s)	Avg wait (ms)	Waits /txn
enq: US - contention	921,506	0	964	1	1,453.48
DFS lock handle	836,342	3	487	1	1,319.15
control file sequential read	6,169	0	15	2	9.73
enq: TA - contention	15,894	0	10	1	25.07
log file parallel write	5,360	0	3	1	8.45
control file parallel write	1,239	0	3	2	1.95
process startup	62	0	2	32	0.10
wait for scn ack	1,722	0	2	1	2.72
gcs log flush sync	1,473	0	1	1	2.32
db file parallel write	547	0	1	2	0.86
...					

10.9. The “SQL ordered by Gets” Section

Buffer Gets	Executes	Gets per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Text
294,966,379	13,943	21,155.16	22.32	1,225.12	1,921.65	160xb7	SELECT COUNT(DISTINCT itm_id) ...
275,674,541	12,704	21,699.82	20.86	1,084.82	1,819.94	7abc9jc	SELECT a.*, CASE (SELECT tmc...
152,046,289	249	610,627.67	11.51	1,011.13	1,568.89	e4xxyc	SELECT c.ctg_id, c.ctg_level,...
124,329,661	1,990	62,477.22	9.41	1,151.67	1,984.38	5uz4x7f	SELECT items.*, usr_screen_na...
...							

10.10. The “SQL ordered by Reads” Section

Physical Reads	Executes	Reads per Exec	%Total	CPU Time (s)	Elapsed Time (s)	SQL Id	SQL Text
39,113,298	1	39,113,298.00	8,984.71	1,371.70	6,330.01	7abc9jc	DECLARE job BINARY_INTEGER := ...
95,886	1	95,886.00	22.03	108.67	508.08	e4xxyc	SELECT itm_id, itm_title, it...
53,464	1	53,464.00	12.28	28.82	132.57	160xb7	select /*+ cursor_sharing_exac...
42,617	1	42,617.00	9.79	28.63	114.52	5uz4x7f	select /*+ cursor_sharing_exac...
...							

10.11. The “Instance Activity Stats” Section

Statistic	Total	per Second	per Trans
CPU used by this session	1,052,252	291.89	1,659.70
CPU used when call started	919,795	255.14	1,450.78
CR blocks created	2,832	0.79	4.47
Cached Commit SCN referenced	10,566,224	2,930.99	16,665.97
Commit SCN cached	63	0.02	0.10
DB time	3,260,955	904.56	5,143.46
DBWR checkpoint buffers written	4,880	1.35	7.70
DBWR checkpoints	34	0.01	0.05
DBWR fusion writes	996	0.28	1.57
DBWR object drop buffers written	802	0.22	1.26
...			

10.12. The “Tablespace IO Stats” Section

Tablespace	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
DATA	175,519	49	3.62	1.05	200	0	802	17.86
INDX	118,530	33	1.01	1.00	3	0	6	3.33
TEMP	12,781	4	1.04	3.48	1,539	0	53	0.75
SYSAUX	5,647	2	3.07	1.03	4,430	1	222	1.76
SYSTEM	771	0	3.68	1.44	273	0	0	0.00
UNDOTBS1	36	0	5.56	1.00	374	0	12	0.83
TEMP1	26	0	5.00	3.73	12	0	0	0.00



10.13. The “File IO Stats” Section

Table space	Filename	Reads	Av Reads/s	Av Rd(ms)	Av Blks/Rd	Writes	Av Writes/s	Buffer Waits	Av Buf Wt(ms)
DATA	+DATA_GRP1/aux/datafile/data366	5,767	2	3.87	1.03	0	0	3	26.67
DATA	+DATA_GRP1/aux/datafile/data.367	7,244	2	3.55	1.06	2	0	3	6.67
DATA	+DATA_GRP1/aux/datafile/data.368	7,820	2	3.58	1.06	25	0	149	3.09
DATA	+DATA_GRP1/aux/datafile/data.369	7,923	2	3.59	1.04	27	0	2	15.00
DATA	+DATA_GRP1/aux/datafile/data.370	7,779	2	3.46	1.05	10	0	71	28.31
DATA	+DATA_GRP1/aux/datafile/data.371	7,816	2	3.51	1.10	0	0	1	320.00
DATA	+DATA_GRP1/aux/datafile/data.372	5,781	2	3.36	1.05	2	0	1	0.00
DATA	+DATA_GRP1/aux/datafile/data.373	5,825	2	3.29	1.04	8	0	1	0.00
DATA	+DATA_GRP1/aux/datafile/data.374	6,370	2	3.44	1.03	26	0	5	48.00
DATA	+DATA_GRP1/aux/datafile/data.375	6,390	2	3.26	1.02	35	0	67	24.03
...									

10.14. The “Buffer Pool Advisory” Section

P	Size for Estimate (M)	Size Factr	Buffers for Estimate	Est Physical Read Factor	Estimated Physical Reads
D	832	0.10	100,360	11.53	56,260,876
D	1,664	0.20	200,720	7.00	34,185,957
D	2,496	0.30	301,080	5.55	27,088,343
D	3,328	0.40	401,440	4.51	22,026,575
D	4,160	0.50	501,800	3.72	18,148,391
D	4,992	0.60	602,160	3.02	14,739,184
D	5,824	0.69	702,520	2.40	11,732,528
D	6,656	0.79	802,880	1.86	9,056,047
D	7,488	0.89	903,240	1.39	6,781,948
D	8,320	0.99	1,003,600	1.03	5,003,716
D	8,384	1.00	1,011,320	1.00	4,881,285
D	9,152	1.09	1,103,960	0.72	3,516,979
D	9,984	1.19	1,204,320	0.58	2,808,241
D	10,816	1.29	1,304,680	0.53	2,584,250
D	11,648	1.39	1,405,040	0.50	2,428,468
D	12,480	1.49	1,505,400	0.49	2,372,076
D	13,312	1.59	1,605,760	0.48	2,335,654
D	14,144	1.69	1,706,120	0.47	2,292,808
D	14,976	1.79	1,806,480	0.46	2,258,187
D	15,808	1.89	1,906,840	0.45	2,217,353
D	16,640	1.98	2,007,200	0.44	2,139,568

10.15. The “PGA Memory Advisory” Section

PGA Target Est (MB)	Size Factr	W/A MB Processed	Estd Extra W/A MB Read/ Written to Disk	Estd PGA Cache Hit %	Estd PGA Overallloc Count
192	0.13	957,939.56	1,268,668.03	43.00	15,646
385	0.25	957,939.56	1,165,321.86	45.00	13,150
769	0.50	957,939.56	10,942.56	99.00	0
1,154	0.75	957,939.56	10,753.92	99.00	0
1,538	1.00	957,939.56	5,483.60	99.00	0
1,846	1.20	957,939.56	5,031.57	99.00	0
2,153	1.40	957,939.56	5,031.57	99.00	0
2,461	1.60	957,939.56	5,031.57	99.00	0
2,768	1.80	957,939.56	5,031.57	99.00	0
3,076	2.00	957,939.56	5,031.57	99.00	0
4,614	3.00	957,939.56	5,031.57	99.00	0
6,152	4.00	957,939.56	5,031.57	99.00	0
9,228	6.00	957,939.56	5,031.57	99.00	0
12,304	8.00	957,939.56	5,031.57	99.00	0

10.16. The “Enqueue Activity” Section

Enqueue Type (Request Reason)	Requests	Succ Gets	Failed Gets	Waits	Wt Time (s)	Av Wt Time(ms)
US-Undo Segment	624,388	624,388	0	483,550	1,008	2.08
PS-PX Process Reservation	66,536	61,098	5,432	10,406	23	2.19
WF-SWRF Flush	11	11	0	4	14	3,430.75
TA-Instance Undo	10,583	10,583	0	8,216	10	1.28
TM-DML	2,337	2,337	0	22	1	34.18
HW-Segment High Water Mark	5,283	5,283	0	181	0	1.30
RO-Multiple Object Reuse (fast object reuse)	183	183	0	34	0	6.79
TX-Transaction (index contention)	3	3	0	3	0	65.33
CF-Controlfile Transaction	7,310	7,310	0	91	0	1.44
SS-Sort Segment	184	184	0	41	0	1.98



10.17. The “Shared Pool Advisory” Section

Shared Pool Size(M)	SP Size Factr	Est LC Size (M)	Est LC Mem Obj	Est LC Time Saved (s)	Est LC Time Saved Factr	Est LC Load Time (s)	Est LC Load Time Factr	Est LC Mem Obj Hits
704	0.49	145	11,741	89,956	1.00	1,007	1.00	12,861,749
848	0.60	166	13,510	89,956	1.00	1,007	1.00	12,861,773
992	0.70	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,136	0.80	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,280	0.90	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,424	1.00	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,568	1.10	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,712	1.20	166	13,510	89,956	1.00	1,007	1.00	12,861,773
1,856	1.30	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,000	1.40	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,144	1.51	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,288	1.61	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,432	1.71	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,576	1.81	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,720	1.91	166	13,510	89,956	1.00	1,007	1.00	12,861,773
2,864	2.01	166	13,510	89,956	1.00	1,007	1.00	12,861,773

10.18. The “SGA Memory Summary” Section

SGA regions	Size in Bytes
Database Buffers	8,791,261,184
Fixed Size	1,342,968
Redo Buffers	2,097,152
Variable Size	1,691,058,696

10.19. The “SGA breakdown difference” Section

Pool	Name	Begin value	End value	% Diff
java	free memory	160,720,192	160,720,192	0.00
java	joxlod exec hp	6,769,728	6,769,728	0.00
java	joxs heap	282,240	282,240	0.00
large	OSM map operations hash t	589,824	589,824	0.00
large	PX msg pool	223,952	223,928	-0.01
large	free memory	15,963,440	15,963,464	0.00
shared	KGLS heap	8,324,744	8,587,672	3.16
shared	KQR L PO	5,508,648	5,676,720	3.05
shared	KQR L SO	628,816	653,392	3.91
...				
shared	type object de	3,008,560	3,015,096	0.22
	buffer_cache	8,791,261,184	8,791,261,184	0.00
	fixed_sga	1,342,968	1,342,968	0.00
	log_buffer	2,097,152	2,097,152	0.00

10.20. The “init.ora Parameters” Section

Parameter Name	Begin value	End value (if different)
cluster_database	TRUE	
cluster_database_instances	4	
compatible	10.1.0.4.0	
db_block_size	8192	
db_file_multiblock_read_count	4	
java_pool_size	167772160	
job_queue_processes	10	
open_cursors	2000	
optimizer_index_cost_adj	10	
parallel_adaptive_multi_user	FALSE	
parallel_execution_message_size	2152	
parallel_max_servers	8	
pga_aggregate_target	1612709888	
processes	1000	
sga_target	10485760000	
undo_management	AUTO	
undo_retention	21600	
undo_tablespace	UNDOTBS1	

11. Conclusions

AIX NMON and Oracle Statspack are powerful performance analysis tools for Oracle DB applications environments that run on IBM System p servers. These tools can be used to quickly identify potential system or application bottlenecks that warrant further analysis. This paper has provided numerous examples of NMON and Statspack output and discussed a number of commonly seen performance/tuning issues.

12. References

- [1] NMON_Analyser User Guide for V3.0, Stephen Atkins, IBM Advanced Technical Sales Support, UK
- [2] “AIX 5 performance series: CPU monitoring and tuning” paper, by Wayne Hung, Lee Cheng and Matthew Accapadi, IBM
- [3] “Understanding IBM @server Performance and Sizing” Redbook, by Diana Gfroerer, Nigel Trickett, Tatsuhiko Nakagawa and Ravi Mani, IBM Publication # SG24-4810
- [4] “Database Performance Tuning on AIX” Redbook, Budi Darmawan, Gary Groenewald, Allan Irving, Sergio Henrique Soares Monteiro, Keirnan M. Snedeker, IBM Publication # SG24-5511
- [5] “AIX 5L Performance Tools Handbook” Redbook, by Budi Darmawan, Charles Kamers, Hennie Pienaar, Janet Shiu, IBM Publication # SG24-6039
- [6] “Advanced POWER Virtualization on IBM System p5” Redbook, Scott Vetter, Annika Blank, Paul Kiefer, Carlos Sallave Jr, Gerardo Valencia, Jez Wain and Armin M. Warda, IBM Publication # SG24-7940
- [7] “Advanced POWER Virtualization on IBM eserver p5 Servers: Architecture and Performance Considerations” Redbook, Scott Vetter, Ben Gibbs, Dr. Balaji Atyam, Frank Berres, Bruno Blanchard, Lancelot Castillo, Pedro Coelho, Nicolas Guerin, Lei Liu, Cesar Diniz Maciel, Dr. Carlos Sosa and Ravikiran Thirumalai, IBM Publication # SG24-5768
- [8] “What is Statspack and where are the READMEs?”, Oracle Metalink note # 149115.1
- [9] “How to Integrate Statspack with EM 10G”, Oracle Metalink note # 274436.1
- [10] “Installing and Configuring Statspack Package”, Oracle Metalink note # 149113.1
- [11] “Gathering a Statspack snapshot”, Oracle Metalink note # 149121.1
- [12] “Creating a Statspack performance report”, Oracle Metalink note # 149124.1
- [13] “Performance Tuning Using 10g Advisors and Manageability Features”, Oracle Metalink note # 276103.1
- [14] “WAITEVENT: "buffer busy waits" Reference Note”, Oracle Metalink note # 34405.1
- [15] “FAQ about Detecting and Resolving Locking Conflicts”, Oracle Metalink note # 15476.1
- [16] “What are Latches and What Causes Latch Contention”, Oracle Metalink note # 22908.1
- [17] “Database Initialization Parameters for Oracle Applications 11i”, Oracle Metalink note # 216205.1
- [18] “Oracle Architecture and Performance Tuning on AIX”, IBM Techdocs document id # WP100657

13. Acknowledgements

Several people provided technical input and guidance for this paper, including:



- Rebecca Ballough, Oracle Solutions Team, ATS Americas
- Ralf Schmidt-Dannert, Oracle Solutions Team, ATS Americas