T.C.

BAHÇEŞEHİR ÜNİVERSİTESİ

IRIS: DEVELOPMENT OF A TOOL FOR PERFORMANCE MONITORING AND TREND ANALYSIS OF INFORMATION TECHNOLOGY INFRASTRUCTURE

Master's Thesis

HAKAN HALİSÇELİK

İSTANBUL, 2009

T.C.
BAHÇEŞEHİR ÜNİVERSİTESİ

THE INSTITUTE OF SCIENCE
INFORMATION TECHNOLOGIES

IRIS: DEVELOPMENT OF A TOOL FOR PERFORMANCE MONITORING AND TREND ANALYSIS OF INFORMATION TECHNOLOGY INFRASTRUCTURE

Master's Thesis

HAKAN HALİSÇELİK

Supervisor: ASST. PROF. DR. ORHAN GÖKÇÖL

İSTANBUL, 2009

T.C.

BAHÇEŞEHİR ÜNİVERSİTESİ
INSTITUTE OF SCIENCE
INFORMATION TECHNOLOGY

Name of the Thesis: Iris: Development of A Tool For Performance Monitoring And Trend Analysis of Information Technology Infrastructure
Name/Last Name of the Student: Hakan Halisçelik
Date of Thesis Defense: 22 January 2009

The thesis has been approved by the Institute of Science.

Prof. Dr. Bülent ÖZGÜLER
Director

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Orhan GÖKÇÖL
Program Coordinator

This is to certify that we have read this thesis and that we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Science.

Examining Committee Members	Signature
Asst. Prof. Dr. Orhan GÖKÇÖL	
Asst. Prof. Dr. M. Alper TUNGA	
Asst. Prof. Dr. Yalçın ÇEKİÇ	

ACKNOWLEDGEMENT

I would like to thank my family, for their patience and confidence, while I was completing this work.

I would like to express my sincere thanks to my project advisor **Asst. Prof. Dr. Orhan GÖKÇÖL** for his valuable guidance.

I would also like to thank to my executive, Uğur Erduğrul, for his sensibility and assistance.

ABSTRACT

IRIS:
DEVELOPMENT OF A TOOL FOR PERFORMANCE MONITORING AND TREND ANALYSIS OF INFORMATION TECHNOLOGY INFRASTRUCTURE

Hakan, Halisçelik

Information Technology

Supervisor: Asst. Prof. Dr. Orhan Gökçöl

January, 2009, 69 Pages

Performance monitoring is one of the most important part of server administration. Performance monitoring includes collecting data, analyzing data and making future considerations based on performance trends. Not only there will be huge amount of data, but performance metrics also vary between different UNIX versions and Windows servers.

The aim of this study is to build a common tool for both UNIX and Windows servers and monitoring health of servers. The tool provides a graphical representation of the data, reports potential bottlenecks on the machine. As a result of this study every server can be watched for possible bottlenecks and also performance improvements can be seen clearly after changes applied. Current study showed that, use of Iris for monitoring server's loads helps admins to diagnose and resolve bottlenecks more certain. With the help of stored data, trend analyses can be done before and after tunings for proving improvements. This thesis also differs from previous works in the way that; this study covers the performance monitoring of both UNIX and Windows systems.

Keywords: Performance Monitoring; UNIX Performance; Windows Performance; Trend Analyses.

ÖZET

IRIS:
BİLGİ TEKNOLOJİLERİ İÇİN PERFORMANS TAKİP VE TREND ANALİZİ ARACI GELİŞTİRME

Hakan, Halisçelik

Bilgi Teknolojileri

Tez Danışmanı: Yrd. Doç. Dr. Orhan Gökçöl

Ocak, 2009, 69 Sayfa

Performans takibi, sistem yöneticiliğinin en önemli kısımlarından biridir. Performans takibinin kapsamı, verinin toplanması, verinin incelenmesi ve gelecek tahminleri yapılmasıdır. Performans takibinin en zor kısımlarından birisi de verinin çokluğu yanında önemli parametrelerin değişik UNIX versiyonları ve Windows versiyonları arasında değişmesidir.

Bu çalışmanın amacı bütün sunucu platformları için ortak bir araç geliştirerek, takibin kolaylaşmasını ve sürekli olmasını sağlamaktır. Geliştirilen bu araç grafiksel olarak performans bilgilerini göstererek olası kaynak sıkışmalarının farkına varılmasını sağlamaktadır. Bu çalışma ile elde edilen faydalardan biri de, tüm sunucuların performans bilgileri izlenebilir olması ve sistemlerde yapılan her türlü değişikliğin sonuçlarının açık olarak görülebilmesidir. Bu çalışma ile şunu gördük ki, sistemlerin anlık durumlarının izlenmesi ile sorunların tespiti ve çözülmesi daha net olmaktadır. Saklanan veri ile de çalışma öncesi ve sonrasını karşılaştırıp performans iyileştirmelerini gösterebilmektedir. Ayrıca bu tez kapsamında, daha önce yapılan çalışmalardan farklı olarka UNIX ve Windows performansları aynı anda izlenmektedir

Anahtar Kelimeler: Performans Takibi; UNIX Performansı; Windows Performansı; Trend Analizi.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

American Standard Code for Information Interchange	:	ASCII
Asynchronous Input and Output	:	AIO
Available Virtual Memory	:	AVM
Bourne Shell	:	SH
C Shell	:	CSH
Central Processing Unit	:	CPU
Cluster Administration using Relational Databases	:	CARD
Common Gateway Interface	:	CGI
Concurrent Input and Output	:	CIO
Database	:	DB
Demilitarized Zone	:	DMZ
Disaster Recovery	:	DR
File System	:	FS
File System Block Size	:	Agblksize
Graphical User Interface	:	GUI
Information Technology	:	IT
Input and Output	:	I/O
Internet Protocol	:	IP
Input and Output	:	I/O
Kilobit Per Second	:	KBPS
Korn Shell	:	KSH
Logical Volume Manager	:	LVM
Page Frame Table	:	PFT
Paging Space	:	PS
Random Access Memory	:	RAM
Read Write Mode	:	RW
Relational Database Management System	:	RDMS
Round Robin Database	:	RRD
Service Queue Full	:	SQFULL

Storage Area Network	:	SAN
System Activity Reporter	:	SAR
The System Administrator’s Cockpit	:	SATOOL
Windows Performance Class	:	WPC
Virtual Memory Manager	:	VMM

1. INTRODUCTION

1.1 SCOPE OF THIS WORK

Performance analysis and performance monitoring are very critical responsibilities for Information Technology. Performance management which contains monitoring, analysis and tuning is rarely considered when performance is good. However, when user is getting bad performance and bad response times, an inability to diagnose and resolve performance problems becomes a major problem. Primary reason for this inability is enough performance data may not have collected during times of good performance. The lack of system baselines could result in an inability to understand what system components are behaving differently since performance has degraded. Because, performance can be bad for all time since user is to rise against admins. So, it is difficult to determine when the system is in normal state or certify that tunings made to the system has any effect on performance.

Other disadvantages of the insufficient performance analysis experience include the poor knowledge in performance metrics and the inability to apply performance tuning to achieve a maximum performing machine (Speier, 2005). In general when there is no complains about performance or when there is no problem on working applications, making system performance tunings or changing variables are strictly forbidden. Furthermore, without enough performance management, poor performance of machines effect IT management and it will cause users to blame the poor performance of applications. This could affect divisions beyond IT that utilize the system.

A UNIX platform can be divided into four performance related components; memory, input/output (Disks), central processing units (CPU), and network (SAN and IP Network). Each component is implemented with totally different algorithms on different UNIX implementations for performance management. For example memory management is fully different in SUN Solaris and IBM AIX. Additionally, each set of algorithms was designed for different operating systems which are working with different hardware. So in a case of problem, it must be considered that which components are performing poorly on a given platform, based on inspecting the output of a set of performance metrics. Not only is this conclusion is difficult to gain, but also the rules for performance management differs between operating systems (Speier, 2005).

A UNIX platform also includes command based performance monitoring utilities. These programs report detailed metrics on nearly all components. For example Vmstat reports statistics relative to server virtual memory. Although this utility is available in all UNIX platforms, the reported data differs over the various operating system versions. Another example is SAR utility. SAR generated data is specific to not only to a particular version of UNIX, but also version and release of operating system.

When this additional level of complexity is considered, performance analysis and monitoring across multiple platforms becomes very difficult. (monitortools, 2009) As a result, providing a common performance analysis and monitoring framework in a homogenous UNIX environment can be a solution for different UNIX versions.

While analyzing the performance of a UNIX server, system administrators will mostly use scheduling utility “cron” to automate performance data collection. The first disadvantage of this approach is that unnecessary information appears in each execution of the performance reporting commands. For example, the title and header fields will be captured for each instance of execution. To eliminate these unnecessary lines advance UNIX scripts must be developed which require advance knowledge. But a common framework that collects data from all defined machines would be so helpful for analysis. Because it already collects data and only need is to define analysis timescale.

Once data collection has ended and performance metrics are collected, system admin can sift collected data for useful information. This is difficult because of the amount of data produced by performance monitoring commands between timescale. The general recommended execution of each command in UNIX is once per minute. As a result, the administrator will view 1,440 records for each component of system performance on a single server each day. In addition to the large overhead required for this approach, it is difficult for the administrator to correlate events from other performance components. For example, a value in one metric could be signal of problem, but combined with values from other metrics could indicate a critical performance issue.

Using this approach to correlate data after it has been collected does not provide real-time analysis. Live production environments require a more proactive monitoring solution for performance issues. In conclusion, when all these obstacles and disadvantages described here are considered, this may not be the best approach for performance analysis.

As a result, as a priori it is considered to build a common framework to monitor major UNIX versions which are SUN Solaris, IBM Aix, Red Hat and HP Tru64. This framework also monitors all the major metrics of Windows based machines. The aim of adding windows servers to this project are to show how easy to integrate other types to Operating Systems to monitoring system.

1.2 ADVANTAGES OF CONTINUOUS PERFORMANCE MONITORING

System performance monitoring are activities performed by a system administrator to make sure system is running with no jam in resources. This monitoring must be continuously for the health of operating environments.

Below items are the advantages of continuous system performance monitoring; (IBM Information Center, 2008)

i. Periodically obtaining performance related information from operating system
ii. Storing the information for future use in problem diagnosis
iii. Displaying the information for the benefit of the system administrator

iv. Detecting situations that require additional data collection or responding to directions from the system administrator to collect such data, or both
v. Collecting and storing the necessary detail data
vi. Tracking changes made to the system and applications

Although monitoring is very important it is also important to specify the critical points. For example; when monitoring the system paging memory, it is useful to monitor total size of paging space but monitoring the use of paging space is more critical in system.

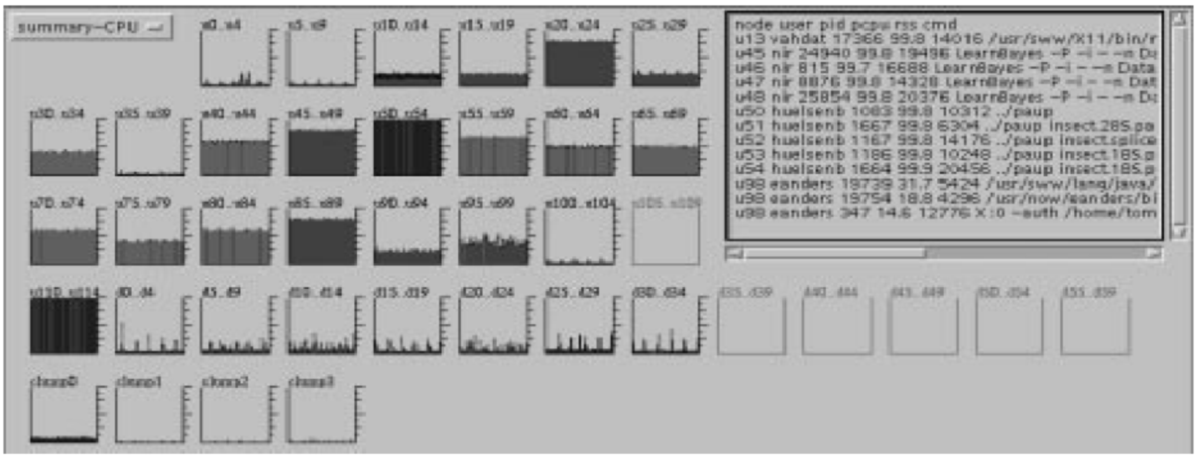
The most important monitoring points are;

i. Response time; response time is the time a system or functional unit takes to react to a given input. (Wikipedia, 2008)
ii. CPU utilization; Percentage of time while the CPU is active
iii. Memory utilization; percentage of real memory which are in use
iv. Paging rate; number of page faults in a given time period
v. Service time; average time that a process spends in execution
vi. Queue length; Number of processes waiting for a resource

1.3 SIMILAR TOOLS

In this part of thesis, some studies done by other researchers and commercial products are summarized by giving emphasis on their findings.

Monitoring a large cluster of cooperating computers requires extensibility, fault tolerance, and scalability (Anderson and Patterson, 1997). “Cluster Administration using Relational Databases” (CARD) system was developed at the University of California, Berkeley. Card system uses MiniSQL to store data and a Java applet GUI to make the system accessible through the web. CARD system monitors the health of nodes by gathering statistics such as CPU and disk usage. Performance Data is gathered at each machine by Perl scripts. The system also uses time-stamp protocols to detect and recover from node failures. In Figure 1.1, a sample of CPU performance of monitored systems can be seen.



```
node user pid pcpu rss cmd
u13 vahdat 17366 99.8 14016 /usr/sww/X11/bin/r
u45 nir 24940 99.8 19496 LearnBayes -P -i - -n Di
u46 nir 815 99.7 16688 LearnBayes -P -i - -n Data
u47 nir 8876 99.8 14328 LearnBayes -P -i - -n Dat
u48 nir 25854 99.8 20376 LearnBayes -P -i - -n Di
u50 huelsenb 1083 99.8 10312 ../paup
u51 huelsenb 1667 99.8 6304 ../paup insect.28S.pa
u52 huelsenb 1167 99.8 14176 ../paup insect.splice
u53 huelsenb 1186 99.8 10248 ../paup insect.18S.p
u54 huelsenb 1664 99.9 20456 ../paup insect.18S.p
u98 eanders 19739 31.7 5424 /usr/sww/lang/java/
u98 eanders 19754 18.8 4296 /usr/now/eanders/bi
u98 eanders 347 14.6 12776 X :0 -auth /home/tom
```

Figure 1.1 : CARD CPU Monitoring Sample

Wolski, Spring and Hayes (1999) designed and implemented a performance forecasting service to provide forecasts of dynamically changing performance characteristics from a distributed set of metacomputing resources. They focus on the problem of making short and medium term forecasts of CPU availability on timeshared UNIX systems. They use similar UNIX tools to monitor running system performance like Vmstat and uptime. The benefit of short term forecasting is important to schedule jobs at non-peak hours, and with medium term forecasting future trend analysis can be done easily.

Another approach to system monitoring is the pulsar system (Finkel, 1997). Pulsar uses distributed scripts (pulse monitors) which measure a statistic and determine whether it is within a set of hardcoded limits. The distributed programs then contact a central display server and report the information to the display server. Pulse monitors are expected to run infrequently with a cron-like scheduled tool. The system has the advantage that it can be extended by just adding additional pulse monitors, which requires no modifications to any of the existing programs. It also has the disadvantage that all of the constants in the pulse monitors needs to be configured by the administrator. Pulsar's centralized design is not fault tolerant, and only simple support for external access to updates.

The System Administrator’s Cockpit (Satool) was developed at the University of Colorado, Boulder, is geared towards early detection of problems occurring in groups of machines (Miller, Stirlen, Nemeth, 1993). Each monitored machine runs a SNMP (Simple Network Management Protocol) agent that executes UNIX scripts to gather data. A data collecting server polls the SNMP agents at set intervals and stores the data in a database. A display system written in Tcl/Tk (Tcl is a scripting language created by John Ousterhout and Tk provides a number of widgets commonly needed to develop desktop applications(Wikipedia, 2008)) provides a GUI for viewing data, checking data for alarm conditions, and interacting with the user. Satool is developed with scalability and uses a hierarchical diagram for displaying host data. Extending Satool involves making simple code changes to the SNMP agent, the data collecting server, and the display system.

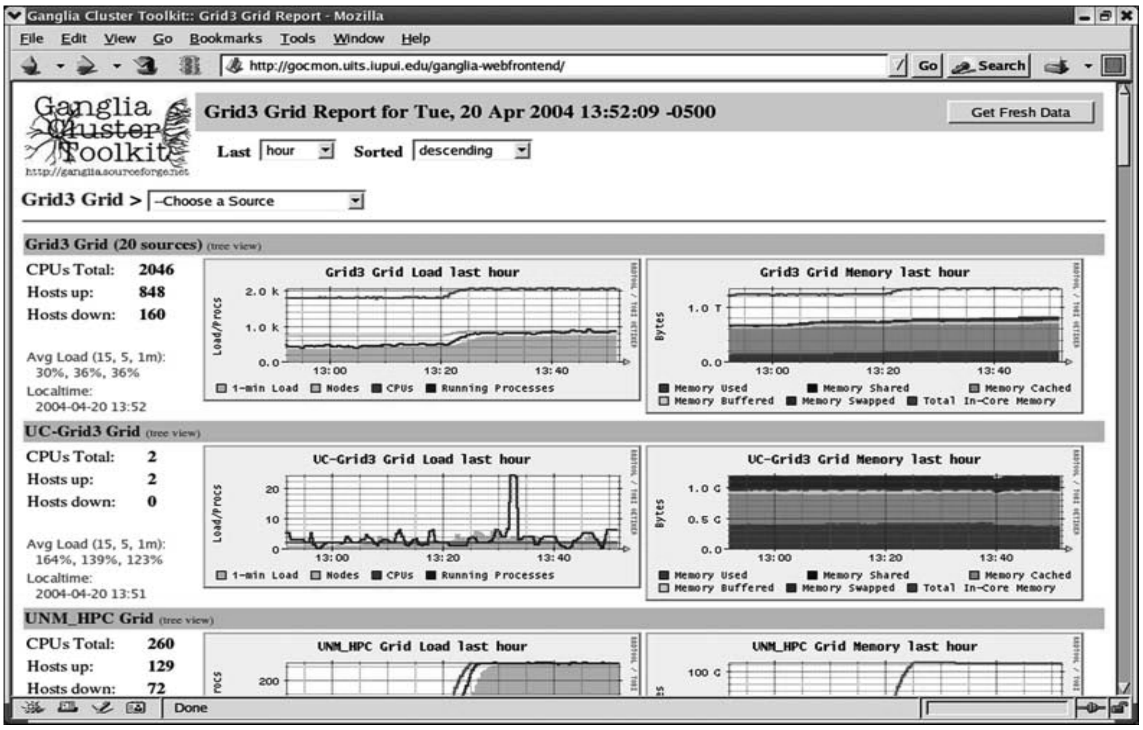
Windows systems performance monitoring tool WatchTower was developed by Knop, Dinda and Schopf (2001). WatchTower has overheads similar to those of Microsoft's Perfmon tool, but it is easily embedded into other software. WatchTower provides easy access to raw performance counters and displays these values in graphs.

A number of other systems (Simonson, 1991), (Shipley and Wang, 1991), (Apisdorf, Claffy, Thompson and Wilder, 1996) show variants on the systems described above. Some of them have very complicated subsystems for statistics gathering, and they vary on whether the gathering happens from a single node, or happens on remote nodes and is sent to a single node. A few of them provide some form of notification other than someone looking at the values on a screen. As a group, they therefore have a similar set of problems to the systems described previously.

Commercial system monitoring packages are also available. Most of these packages are large products that concentrate on specific areas such as network management rather than system management as a whole. A few examples follow.

One of the most famous tools is Ganglia tool. Ganglia is an open-source project which was started at the University of California with the Berkley Millennium Project. (Millennium project is a cluster project to link small computers to make a super computer in university campus) It is a scalable distributed monitoring system for high-performance computing systems such as clusters and grids (Wikipedia, 2008). Ganglia provide a web based front end to display real time data for clusters and each system in a cluster. A multithreaded client process runs on each node to collect and communicate the host situation in real time. So this was the first disadvantage. Installing monitor client applications on every server is difficult and risky. Also monitoring and maintaining every client application is very time consuming process.

Ganglia uses widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRD tool for data storage and visualization. RRD tool is also very widely used because of its easy implementation and extremely gifted. It uses Round Robin algorithm to store data to its own data stores. And it also has ability to show stored data with graphical views. Ganglia has a graphical interface with dynamic web pages which is a “must have” characteristic for today’s applications. A sample output can be seen in Figure 1.2.


Ganglia Cluster Toolkit:: Grid3 Grid Report - Mozilla
File Edit View Go Bookmarks Tools Window Help
http://gocmon.uits.iupui.edu/ganglia-webfrontend/
Go
Search
Ganglia Cluster Toolkit
http://ganglia.sourceforge.net
Get Fresh Data
Last hour
Sorted descending
Grid3 Grid >
--Choose a Source
Grid3 Grid Load last hour
Grid3 Grid Memory last hour
UC-Grid3 Grid Load last hour
UC-Grid3 Grid Memory last hour
UNM_HPC Grid Load last hour
UNM_HPC Grid Memory last hour
Load/Procs
Bytes
1-min Load
Nodes
CPUs
Running Processes
Memory Used
Memory Shared
Memory Cached
Memory Buffered
Memory Swapped
Total In-Core Memory
Done

Grid3 Grid Report for Tue, 20 Apr 2004 13:52:09 -0500

Grid3 Grid (20 sources) (tree view)
CPUs Total: 2046
Hosts up: 848
Hosts down: 160
Avg Load (15, 5, 1m): 30%, 36%, 36%
Localtime: 2004-04-20 13:52

UC-Grid3 Grid (tree view)
CPUs Total: 2
Hosts up: 2
Hosts down: 0
Avg Load (15, 5, 1m): 164%, 139%, 123%
Localtime: 2004-04-20 13:51

UNM_HPC Grid (tree view)
CPUs Total: 260
Hosts up: 129
Hosts down: 72

Figure 1.2 : Ganglia Sample Web Page

Source: www.ganglia.info, 2008

By default, Ganglia monitor a set of metrics, including CPU load, memory usage, and network traffic. It also provides a tool called “Gmetric” which enables to extend the set of metrics they monitor. Ganglia can be a powerful tool for cluster administrators who need to monitor the system utilization and health of cluster nodes.

1.4 THESIS ROADMAP

This thesis is covered in four main parts.

First chapter is introduction part. In this part, brief information about performance monitoring, benefits of monitoring and similar monitoring tools are going to given.

In second chapter; the common performance monitoring commands, performance related parameters are going to be showed.

Chapter three includes development phases. Firstly, information about used technology and methods will be given. Then parts of development will be introduced.

Chapter four is a discussion part with a performance monitoring and tuning problem. Benefits of monitoring are going to be shown. This thesis ends with further comments about how it can be better and what are the limitations.

2. UNIX PERFORMANCE MONITORING

In UNIX there are 4 major resource types that need to be monitored. These are;

- CPU
- Memory
- Disk
- Processes.

Although these four elements are important, monitoring and watching CPU and memory in real time is mission critical job. Because a bottleneck or something wrong in system can be easily seen from the usage of CPU or memory. There are several ways of monitoring activity. Best of monitoring tools are;

- Vmstat (Memory)
- SAR (CPU)
- Iostat (Disk)
- Ps (Process)

From now on information about how to use these tools and important values are going to be given for each tool.

2.1 THE VMSTAT COMMAND

The first tool to use is the Vmstat command. Vmstat quickly provides compact information about different system resources and their related performance problems for UNIX cloned Operating Systems (OS). Vmstat command reports statistics about kernel threads (run and wait queue), memory, swap paging, disks, interrupts, system calls, context switches, and CPU activity. (Hayashi et al. 2005) The reported CPU activity is a percentage view of users, system, idle time, and waits for disk I/O.

The Vmstat output is very useful because it gives a good summary of the system resources on a single line. Table 2.1 is a sample output of Vmstat taken from a UNIX server.

Table 2.1 : Vmstat command output

# **vmstat 2 5**

r	b	Avm	Fre	re	Pi	po	fr	Sr	Cy	In	Sy	cs	us	Sy	id	wa
0	0	51696	49447	0	0	0	6	36	0	104	188	65	0	1	97	2
0	0	51698	49445	0	0	0	0	0	0	472	1028	326	0	1	99	0
0	0	51699	49444	0	0	0	0	0	0	471	990	327	0	1	99	0
0	0	51700	49443	0	0	0	0	0	0	473	992	330	0	1	99	0
0	0	51701	49442	0	0	0	0	0	0	469	986	329	0	0	99	0

The reported **most important** fields are: (Hayashi et al. 2005)

i. **Processes;**

r (Run queue) Average number of threads on the run queues of CPU per second. These threads are only waiting for CPU and are ready to run. A high number of run queue doesn't means a system bottleneck. Because small process gets in queue like bigger processes. If this value is always more than 10 this should be considered as a problem. If this value is 0 system is idle. In Table 2.1 the system is also idle.

b (Blocked queue) Average number of blocked threads on CPU queue per second. These threads are waiting for resource like other threats or disk I/O. They can also wait for their memory information which is swapped to disk, to move to the main memory.

ii. **Memory;**

avm Active Virtual Memory (avm) indicates the number of virtual pages which are accessed. In Table 2.1, there are 51701 pages accessed. This makes 25, 24 GB of memory.

fre This indicates the size of the free pages or memory pages. Terminating applications release their memory, and those free memories are added back to the free list. But File system caches are not added back to the free list because they are used by main kernel. In Table 2.1 there are 49442 free 4k pages. This makes 24, 14 GB free memory

iii. Paging Activity;

re	The number of returned memory pages per second. When a process releases its unused memory, re value increases meanwhile.
pi	The number of page-in requests. These pages were paged to paging space (PS) and now paging into memory, because they are required by a process. When a system is paging data from PS to main memory, processes gets slower performance. Because CPU must wait for data before processing the thread (Blocked Queue Value). A high value of pi is a symptom of memory shortage.
po	The number of pages-out. When there is a memory shortage and new processes demands more memory, old pages are paged out to PS by the VMM. They will stay in PS and be paged in if required. If po value is high like pi value, there is also a memory shortage.
fr	Number of pages freed. When the VMM requires memory, page-replacement algorithm runs to scan the Page Frame Table (PFT) to determine which pages to steal. If a page has not been used since the last scan, it can be stolen, because it is not frequently used. These pages are handled in two ways; first VMM can move them to PS because they can be used later on. Second way is if there is no I/O on that page it can be deleted without moving to PS.
sr	Represents pages scanned by the page-replacement algorithm. When page stealing occurs, the pages in memory are scanned to determine which can be stolen. It is important that a memory shortage is occurring. It is always important to consider “Fr;Sr” values together.
cy	This value refers to the number of times the page replacement algorithm completes the cycle through memory for pages to steal. If this value is greater than zero, this means there is a memory shortage. Because it continuously looks for steal able memory. This is as important as Sr value.

iv. CPU usage information;

us (User time) Programs can run in either user mode or system mode. In user mode, the program does not need the resources of the kernel to manage memory or perform computations. It is like unprivileged mode of running processes. If us value is high so users who are running processes should be investigated.

sy (System time) Processes which are running in system mode can use kernel processes and others kernel resources. Processes requiring the use of kernel services must switch to service mode to gain access to the services, such as to open a file or read/write data.

id (CPU idle time) This indicates the percentage of time the CPU is idle without I/O. When the CPU is idle, it has nothing on the run queue.

wa (CPU wait time) CPU idle time while the system has at least one waiting I/O to disk. An I/O causes the process to block until the I/O is complete. Upon completion, it is placed on the run queue. If wa is over 25 percent, this indicates a need to investigate the disk I/O subsystem throughput.

By using given information, Vmstat output (Table 2.2) will be examined for possible performance bottlenecks.

Table 2.2 : Sample Vmstat output

r	b	avm	Fre	Re	pi	po	fr	sr	cy	in	sy	Cs	us	Sy	Id	wa
6	2	9241890	10265	0	0	0	5479	12977	0	4268	75781	27928	31	4	64	1
10	1	9243275	11491	0	0	0	8231	22014	0	5201	160549	32796	47	5	47	1
5	1	9242010	10400	0	0	0	3753	11324	0	5698	87368	30611	36	4	59	2
6	2	9242173	10114	0	0	0	3611	9909	0	7423	86722	26990	43	4	50	3
8	2	9249418	12238	0	0	0	12542	64468	0	6720	123088	23951	42	9	47	2
8	1	9245422	15967	0	0	0	3361	24988	0	7137	109061	26310	38	7	52	2
10	2	9244330	14921	0	0	0	1286	9405	0	6630	124307	24613	28	8	61	3
6	2	9250891	10093	0	0	0	4784	23181	0	6190	101402	20800	38	7	54	2
8	1	9247761	10100	0	0	0	13154	90745	0	7335	128523	53863	39	13	45	2
12	3	9248017	14437	0	0	0	17635	108201	0	9125	125900	49798	32	13	52	4
6	1	9246528	15945	0	0	0	3220	18827	0	6945	118733	29599	31	8	58	3
20	0	9245635	16473	0	0	0	3738	18663	0	6972	155553	34510	37	9	53	2
4	1	9245930	13002	0	0	0	1677	9621	0	7184	229339	27517	26	6	65	3

i. Run queue (r) is between 4 to 20, not bad.
ii. Number of Blocked waiting (b) process is low. No wait for I/O or memory.
iii. There is no Page in (pi) or out (po) so system is not having memory problem.
iv. (fr:sr) ratio, the page steal algorithms are working to find unused or less used memory. As pi value is zero, the memory is being stolen successfully without the need for paging.
v. Us+sys are not more than 50 percent, so there is no CPU shortage.
vi. There is at most 4 percent wa, so disk subsystem is well tuned.

From the output of Vmstat nearly all performance related information about memory and CPU can be seen.

2.1.1 Vmstat VMM Statics

Vmstat's main purpose is system virtual or physical memory monitoring. It can report detailed reports of data about VMM. This also includes the tunable parameters of VMM. Figure 2.1 is an example output of Vmstat VMM statics taken from a server.

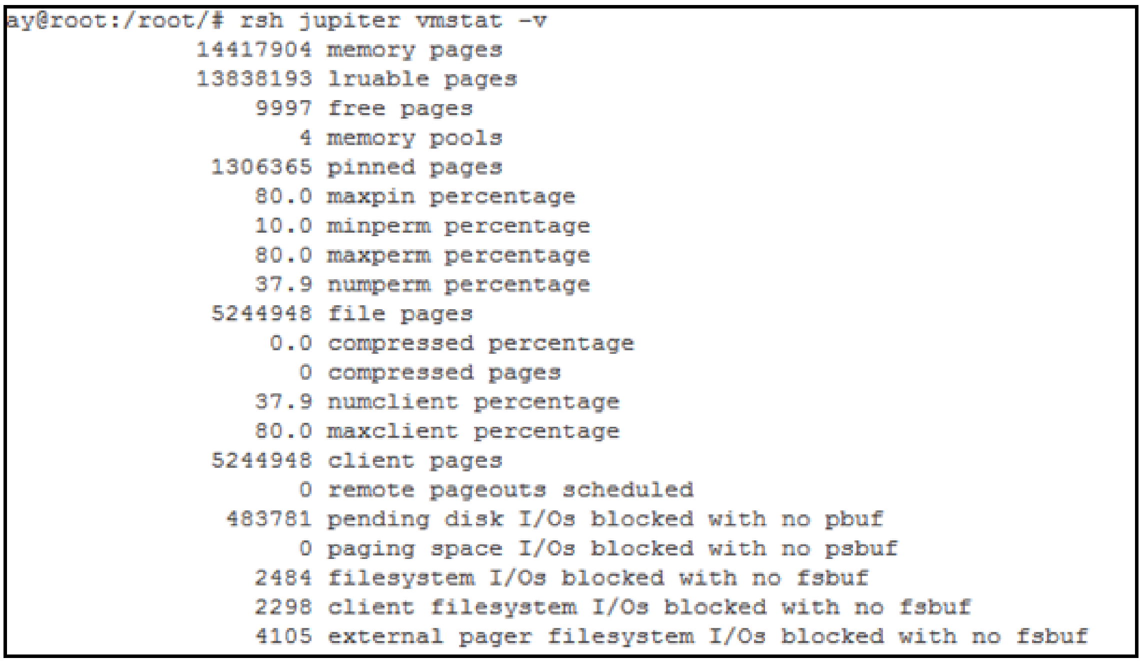

```
ay@root:/root/# rsh jupiter vmstat -v
             14417904 memory pages
             13838193 lruable pages
                 9997 free pages
                    4 memory pools
              1306365 pinned pages
                 80.0 maxpin percentage
                 10.0 minperm percentage
                 80.0 maxperm percentage
                 37.9 numperm percentage
              5244948 file pages
                  0.0 compressed percentage
                    0 compressed pages
                 37.9 numclient percentage
                 80.0 maxclient percentage
              5244948 client pages
                    0 remote pageouts scheduled
               483781 pending disk I/Os blocked with no pbuf
                    0 paging space I/Os blocked with no psbuf
                 2484 filesystem I/Os blocked with no fsbuf
                 2298 client filesystem I/Os blocked with no fsbuf
                 4105 external pager filesystem I/Os blocked with no fsbuf
```

Figure 2.1 : Vmstat VMM Statics Report Sample Output

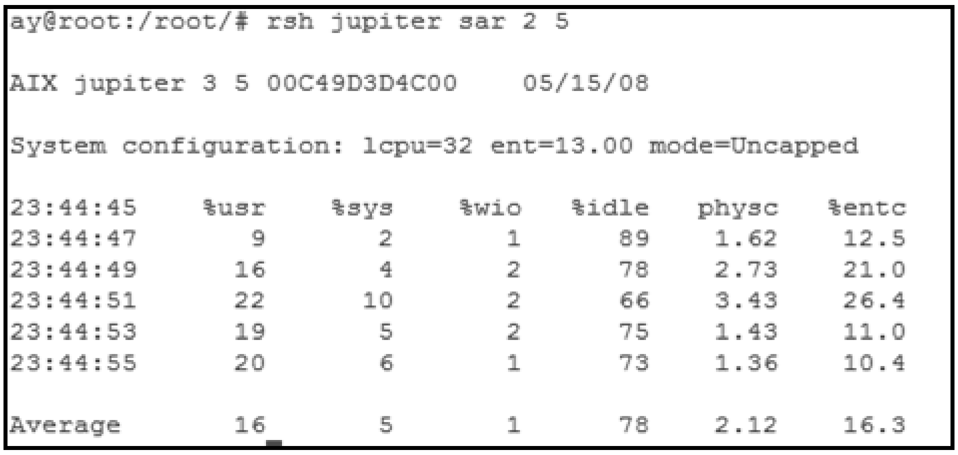
The most important values in Figure 2.1 are;

memory pages	Size of real memory in 4 KB pages. (14417904*4/10^9=55GB)
lruable pages	Number of 4 KB pages considered for replacement.
free pages	Number of free 4 KB pages. (~39mb free memory)

MinPerm, MaxPerm, Numperm, pbuf, psbuf and Fsbuf parameters are some of the most important values. These values will be discussed in last chapter.

2.2 THE SAR COMMAND

The SAR command is used to gather statistical information about system CPU, queuing, paging, file access, and more. When starting to look for a potential performance bottleneck, system admin needs to find out more about how the system uses CPU, memory, and I/O. For these resources information SAR command can be used.



```
ay@root:/root/# rsh jupiter sar 2 5

AIX jupiter 3 5 00C49D3D4C00    05/15/08

System configuration: lcpu=32 ent=13.00 mode=Uncapped

23:44:45    %usr    %sys    %wio   %idle   physc   %entc
23:44:47       9       2       1      89    1.62    12.5
23:44:49      16       4       2      78    2.73    21.0
23:44:51      22      10       2      66    3.43    26.4
23:44:53      19       5       2      75    1.43    11.0
23:44:55      20       6       1      73    1.36    10.4

Average       16       5       1      78    2.12    16.3
```

Figure 2.2 : SAR command output

Figure 2.2 is an example of SAR command for processor usage. Command collects data with a sample in 2 seconds for 5 times. The output consists of user, kernel load, I/O waiting processes load and server idle values. Output is similar to Vmstat. From the output, how much percent of CPU is taken by application, kernel or waiting disk I/O requests can be seen.

With the new features of SAR in IBM AIX environment users can see how much processor power the system is using. It is important in virtual partitions. %entc is also a new feature; it shows that this server has 13 physical processor powers and it uses 16, 3 percent of it (~2.12 processor power). SAR has a lot of features to show server performance, these are shown in Table 2.3.

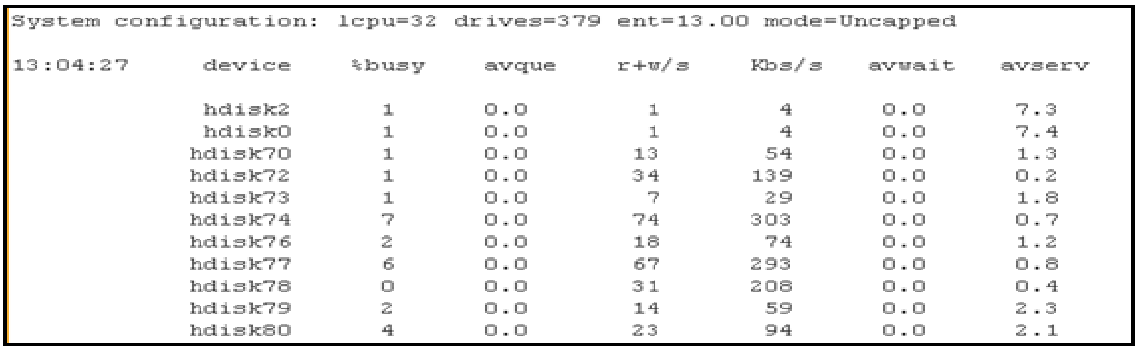
Table 2.3 : SAR Features

Parameter	Information
-a	Checks file access operations
-b	Checks buffer activity
-c	Checks system calls
-d	Checks activity for each block device (Disk)
-g	Checks page out and memory freeing
-k	Checks kernel memory allocation
-m	Checks inter process communication
-p	Checks swap and dispatch activity
-q	Checks queue activity
-r	Checks unused memory
-u	Checks CPU utilization
-nv	Checks system table status
-w	Checks swapping and switching volume
-y	Checks terminal activity
-A	Reports overall system performance (same as entering all options)

Source: www.softpanorama.org, 2008. System Activity Reporter (SAR)

2.2.1 Monitoring Disk Activity with SAR

One of the most needed SAR parameter is “-d” which monitors the disk activity.



```
System configuration: lcpu=32 drives=379 ent=13.00 mode=Uncapped

13:04:27      device     %busy     avque     r+w/s     Kbs/s    avwait    avserv

              hdisk2       1        0.0        1         4        0.0       7.3
              hdisk0       1        0.0        1         4        0.0       7.4
             hdisk70       1        0.0       13        54        0.0       1.3
             hdisk72       1        0.0       34       139        0.0       0.2
             hdisk73       1        0.0        7        29        0.0       1.8
             hdisk74       7        0.0       74       303        0.0       0.7
             hdisk76       2        0.0       18        74        0.0       1.2
             hdisk77       6        0.0       67       293        0.0       0.8
             hdisk78       0        0.0       31       208        0.0       0.4
             hdisk79       2        0.0       14        59        0.0       2.3
             hdisk80       4        0.0       23        94        0.0       2.1
```

Figure 2.3 : SAR Disk Activity Sample Output

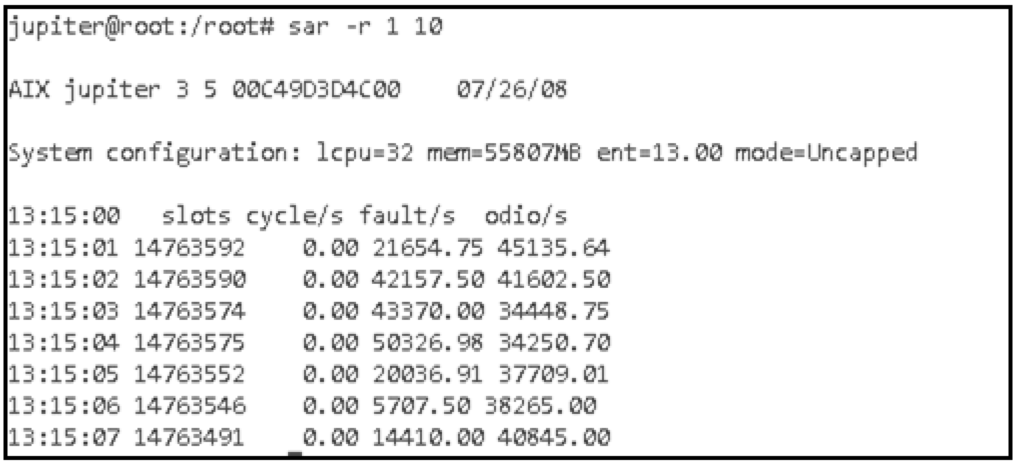
As in Figure 2.3 every defined disk in system can be monitored. It is also possible to get this report for specific disks. Important values are;

%busy	The time while the device was busy.
avque	The average number of requests in the queue
r+w/s	Number of read and write requests per second
blks/s	Number of bytes transferred in 512-byte blocks per second
avwait	The average time requests wait in the queue before it is serviced
avserv	The average service time

As seen from Figure 2.3, there is no disk subsystem bottleneck. For hdisk0 and hdisk2 average service times are very high than the others. These disks are local SCSI disks while the others are fiber SAN disks. For system performance health, it is very important to check disks for bottlenecks.

2.2.2 Monitoring Paging Activity with SAR

Paging activity can be monitored in SAR with "–r" parameter. It is similar to Vmstat command.



```
jupiter@root:/root# sar -r 1 10

AIX jupiter 3 5 00C49D3D4C00    07/26/08

System configuration: lcpu=32 mem=55807MB ent=13.00 mode=Uncapped

13:15:00   slots cycle/s fault/s  odio/s
13:15:01 14763592    0.00 21654.75 45135.64
13:15:02 14763590    0.00 42157.50 41602.50
13:15:03 14763574    0.00 43370.00 34448.75
13:15:04 14763575    0.00 50326.98 34250.70
13:15:05 14763552    0.00 20036.91 37709.01
13:15:06 14763546    0.00 5707.50 38265.00
13:15:07 14763491    0.00 14410.00 40845.00
```

Figure 2.4 : SAR Paging Monitoring Sample Output

The important elements for paging activity at Figure 2.4 are;

slots	Number of free 4096 byte pages on the PS. (Nearly 57 GB PS)
cycle/s	Number of page replacement cycles per second.

fault/s	Number of page faults per second.
odio/s	Number of non-paging disk I/O's per second.

From the above example it can be seen that system is not swapping memory to disk. So there is enough memory. Cycles are 0 so there is no data traffic from swap to memory. Faults are high so memory hit ratio is high which is good for performance.

2.2.3 SAR Performance Data Collection

SAR also has performance data collection feature for specified period of time. Statistical information can be collected by editing the crontab entries of SAR for "adm" user.

```
0 8-17 * * 1-5 /usr/lib/sa/sa1 1200 3 &
5 18 * * 1-5 /usr/lib/sa/sa2 -s 8:00 -e 18:01 -i 3600 -ubcwyaqvm &
```

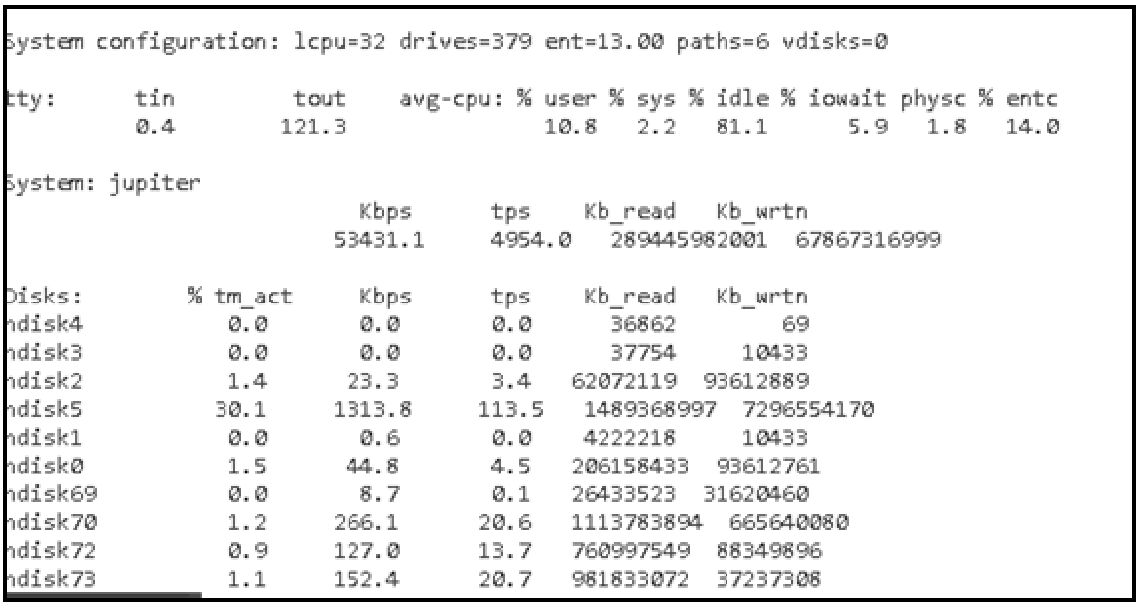
In above example with /usr/lib/sa/sa1 all processor activity is collected for the period of 8:00 to 17:00 for each day between 1st days of week (Monday) to 5th day of week (Friday). Sa1 commands create binary files in the /var/adm/sa directory for each day. This file contains only performance data in SAR data format.

With /usr/lib/sa/sa2 command this data will be converted to ASCII format. This command is scheduled sa2 to work at 18:15 because sa1 command ends at 18:00. It is also possible to collect sa1 data only. When ASCII report is needed, "sar –F /var/adm/sa/sa<needed day>" command converts to readable report.

2.3 IOSTAT COMMAND

The Iostat command is used for monitoring system input/output device (disk) load by observing the time the physical disks are active (linuxcommand, 2008). Iostat generates reports that can be used to tune system disk configuration for better balance the I/O load between physical disks and adapters. The primary purpose of the Iostat tool is to detect I/O bottlenecks by monitoring the disk utilization.

It is useful to run Iostat whether your system is under load or performing normally. This gives a baseline to determine future performance problems with the disk subsystem.



```
ystem configuration: lcpu=32 drives=379 ent=13.00 paths=6 vdisks=0

ty:      tin        tout    avg-cpu: % user % sys % idle % iowait physc % entc
         0.4       121.3             10.8    2.2   81.1      5.9   1.8   14.0

ystem: jupiter
                          Kbps      tps    Kb_read   Kb_wrtn
                        53431.1     4954.0   289445982001  67867316999

isks:          % tm_act     Kbps      tps    Kb_read   Kb_wrtn
disk4            0.0       0.0      0.0      36862        69
disk3            0.0       0.0      0.0      37754     10433
disk2            1.4      23.3      3.4   62072119  93612889
disk5           30.1    1313.8    113.5   1489368997  7296554170
disk1            0.0       0.6      0.0    4222218     10433
disk0            1.5      44.8      4.5   206158433  93612761
disk69           0.0       8.7      0.1   26433523  31620460
disk70           1.2     266.1     20.6   1113783894  665640080
disk72           0.9     127.0     13.7   760997549  88349896
disk73           1.1     152.4     20.7   981833072  37237308
```

Figure 2.5 : Iostat Disk Performance Sample

As seen Figure 2.5, Iostat first gives system processor based performance data, like Vmstat or SAR. Then, there is detailed information about disks.

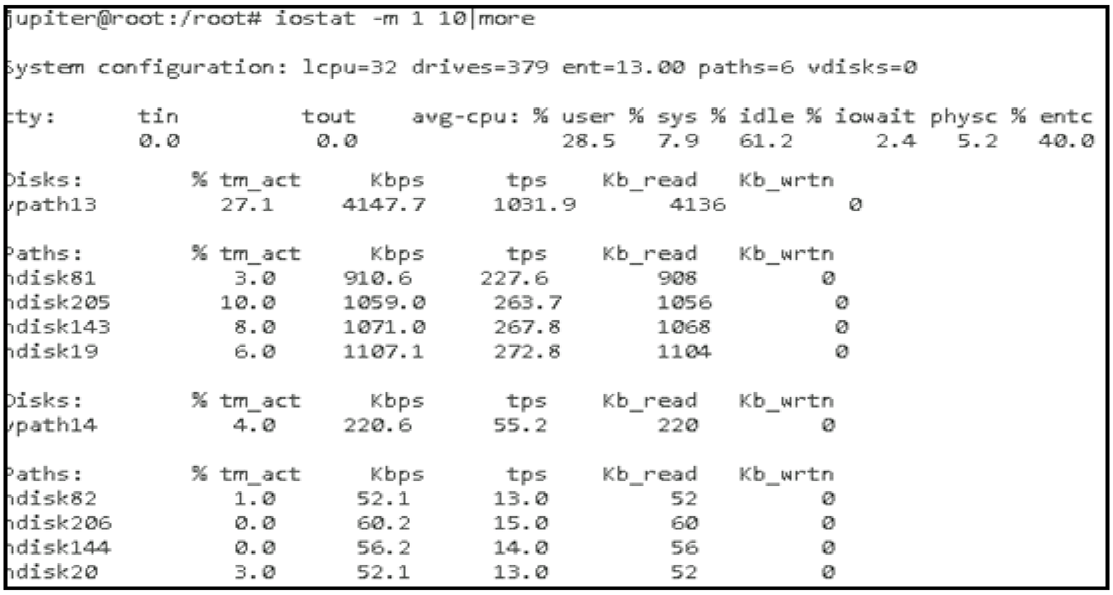
Some important columns of Iostat output in Figure 2.5 are;

%tm_act	Indicates the percentage of time the physical disk was active. This is the primary indicator of a bottleneck. Any %tm_act over 90 percent may be considered a potential bottleneck.
Kbps	Indicates the amount of read or write to the disk in KB per second. The total data written or read from disk information is kb_read or kb_write columns.

From example in Figure 2.5, only hdisk5 hits the 30 percent tm_act. It is not a problem since its below 90 percent. If it was higher than 90 percent which process makes that saturation must be investigated.

2.3.1 Disk Utilization for Multi Pathing

In general servers are connected to SAN with multiple paths for redundancy and load balancing. They are both active backups for each other. In above Figure 2.6, the server is connected to SAN with 4 paths. They are both active-active and load balancing.



```
jupiter@root:/root# iostat -m 1 10|more

System configuration: lcpu=32 drives=379 ent=13.00 paths=6 vdisks=0

tty:      tin        tout    avg-cpu: % user % sys % idle % iowait physc % entc
          0.0         0.0              28.5    7.9   61.2      2.4   5.2   40.0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
vpath13          27.1     4147.7    1031.9      4136         0

Paths:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk81          3.0      910.6     227.6       908        0
hdisk205        10.0     1059.0     263.7      1056         0
hdisk143         8.0     1071.0     267.8      1068         0
hdisk19          6.0     1107.1     272.8      1104         0

Disks:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
vpath14          4.0      220.6      55.2       220        0

Paths:        % tm_act     Kbps      tps    Kb_read   Kb_wrtn
hdisk82          1.0       52.1      13.0        52        0
hdisk206         0.0       60.2      15.0        60        0
hdisk144         0.0       56.2      14.0        56        0
hdisk20          3.0       52.1      13.0        52        0
```

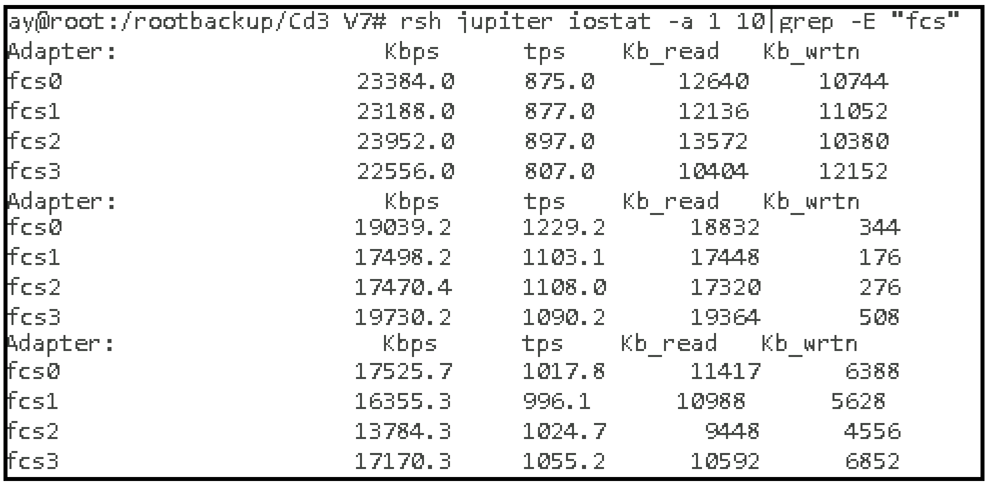
Figure 2.6 : Iostat Multi Pathing Example

As shown in Iostat –m parameter (Figure 2.6); hdisk81, hdisk205, hdisk143 and hdisk19 are the same physical disk in storage system. The I/O to this disk splits into 4 Fiber path and written to same disk. For vpath13 in Figure 2.6, there is a total write I/O 4148kbps and these splits into 4 paths.

With the help of “iostat –m”, a problem with multi pathing or fiber paths can also be seen.

2.3.2 Adapter Throughput Report

Iostat tool can also check the condition and performance of Fiber host adapters. As shown in the Figure 2.7 every Kbps values of Fiber adapters must be nearly same because of the load balancing. If these values are very different from each other there must be something to check.



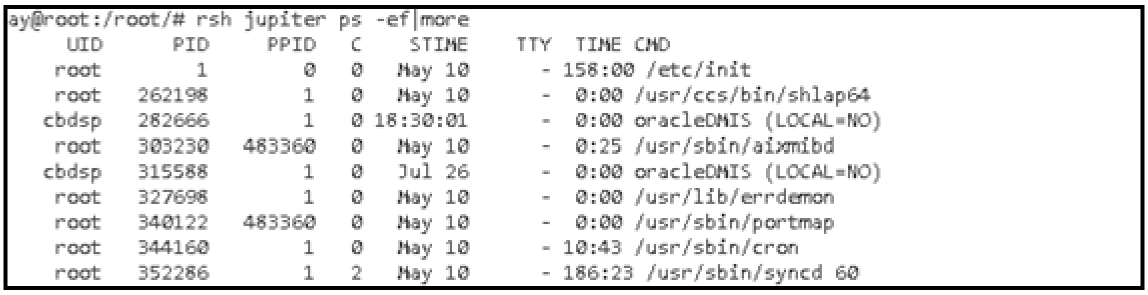
```
ay@root:/rootbackup/Cd3 V7# rsh jupiter iostat -a 1 10|grep -E "fcs"
Adapter:                    Kbps      tps    Kb_read   Kb_wrtn
fcs0                     23384.0     875.0     12640     10744
fcs1                     23188.0     877.0     12136     11052
fcs2                     23952.0     897.0     13572     10380
fcs3                     22556.0     807.0     10404     12152
Adapter:                    Kbps      tps    Kb_read   Kb_wrtn
fcs0                     19039.2    1229.2      18832       344
fcs1                     17498.2    1103.1      17448       176
fcs2                     17470.4    1108.0      17320       276
fcs3                     19730.2    1090.2      19364       508
Adapter:                    Kbps      tps    Kb_read   Kb_wrtn
fcs0                     17525.7    1017.8      11417      6388
fcs1                     16355.3     996.1     10988      5628
fcs2                     13784.3    1024.7       9448      4556
fcs3                     17170.3    1055.2      10592      6852
```

Figure 2.7 : Iostat Host Adapter Throughput Sample

It is also possible to check the condition of Fiber paths by checking the Kb per second value. If there is a problem it should be zero.

2.4 PS COMMAND

Process Status (PS) command makes a list of running processes on the system that can be used to determine; how long a process has been running, how much CPU resource the processes are using and all statics about running processes. It also shows how much memory processes are using, how much I/O a process is performing, the priority and nice values for the process, and who created the process.



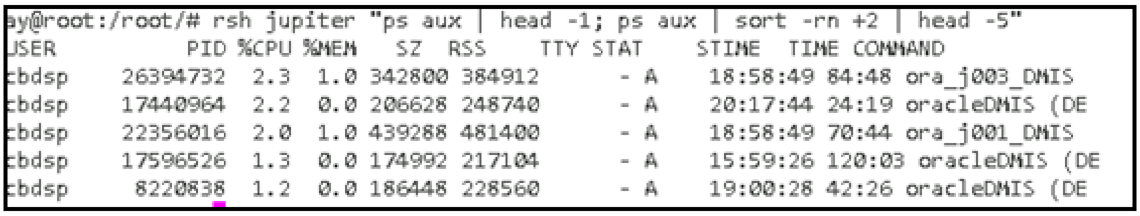
```
ay@root:/root/# rsh jupiter ps -ef|more
     UID      PID     PPID   C    STIME    TTY  TIME CMD
    root        1        0   0   May 10      - 158:00 /etc/init
    root   262198        1   0   May 10      -  0:00 /usr/ccs/bin/shlap64
   cbdsp   282666        1   0 18:30:01      -  0:00 oracleDMIS (LOCAL=NO)
    root   303230   483360   0   May 10      -  0:25 /usr/sbin/aixmibd
   cbdsp   315588        1   0   Jul 26      -  0:00 oracleDMIS (LOCAL=NO)
    root   327698        1   0   May 10      -  0:00 /usr/lib/errdemon
    root   340122   483360   0   May 10      -  0:00 /usr/sbin/portmap
    root   344160        1   0   May 10      - 10:43 /usr/sbin/cron
    root   352286        1   2   May 10      - 186:23 /usr/sbin/syncd 60
```

Figure 2.8 : PS Command Sample Output

Running processes in system can be listed as in Figure 2.8.

2.4.1 Top CPU Processes

As shown in Figure 2.8 all running processes can be listed. PS command can also list processes according to their CPU usage. The Figure 2.9 is an example of listing top CPU consuming processes.



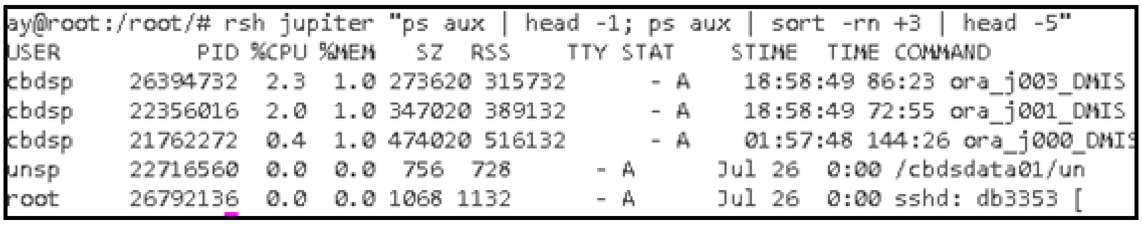
```
ay@root:/root/# rsh jupiter "ps aux | head -1; ps aux | sort -rn +2 | head -5"
JSER          PID %CPU %MEM    SZ  RSS    TTY STAT    STIME  TIME COMMAND
bdsp     26394732  2.3  1.0 342800 384912      - A    18:58:49 84:48 ora_j003_DMIS
bdsp     17440964  2.2  0.0 206628 248740      - A    20:17:44 24:19 oracleDMIS (DE
bdsp     22356016  2.0  1.0 439288 481400      - A    18:58:49 70:44 ora_j001_DMIS
bdsp     17596526  1.3  0.0 174992 217104      - A    15:59:26 120:03 oracleDMIS (DE
bdsp      8220838  1.2  0.0 186448 228560      - A    19:00:28 42:26 oracleDMIS (DE
```

Figure 2.9 : Ps Top CPU Processes Sample Output

(This command is not complicated as seen in Figure 2.9. Another tool called sort is used for sorting. Output is sorted according to second column for CPU usage. If output is sorted for third column, it displays top memory consuming processes.)

2.4.2 Top Memory Processes

With PS command, memory usage of running processes can also gathered. Like in Chapter 2.4.2 output of PS can be sorted according to memory usage. Below Figure 2.10 is the top memory user processes.

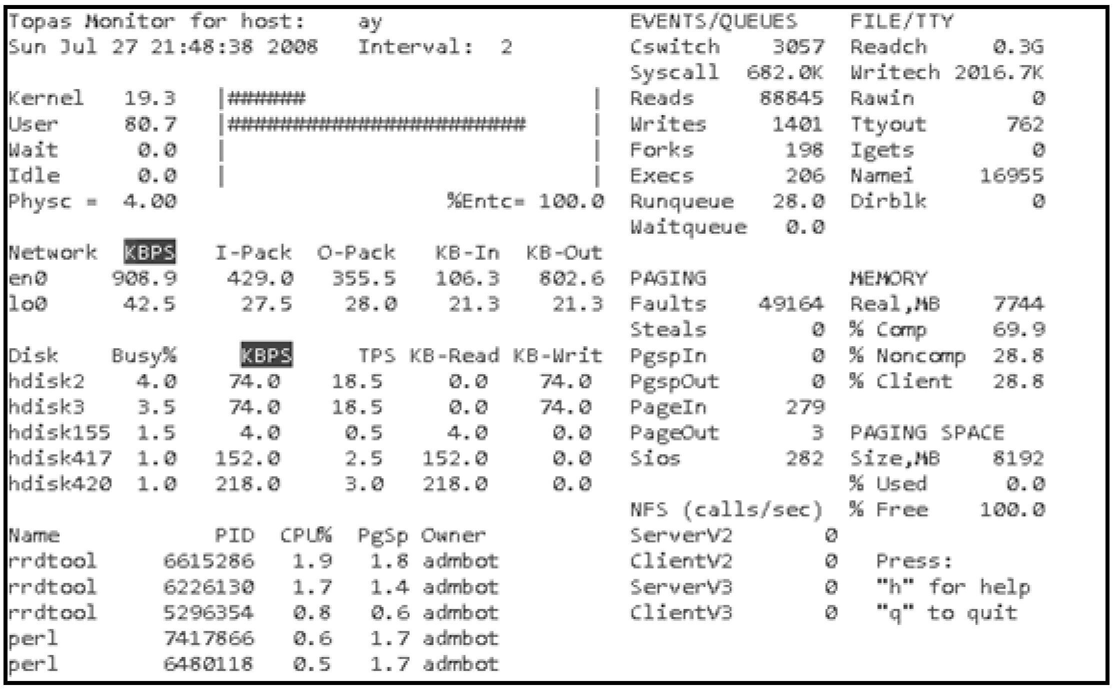


```
ay@root:/root/# rsh jupiter "ps aux | head -1; ps aux | sort -rn +3 | head -5"
USER          PID %CPU %MEM    SZ  RSS     TTY STAT    STIME  TIME COMMAND
cbdsp    26394732  2.3  1.0 273620 315732      - A    18:58:49 86:23 ora_j003_DMIS
cbdsp    22356016  2.0  1.0 347020 389132      - A    18:58:49 72:55 ora_j001_DMIS
cbdsp    21762272  0.4  1.0 474020 516132      - A    01:57:48 144:26 ora_j000_DMIS
unsp     22716560  0.0  0.0  756  728      - A      Jul 26  0:00 /cbdsdata01/un
root     26792136  0.0  0.0 1068 1132      - A      Jul 26  0:00 sshd: db3353 [
```

Figure 2.10 : Ps Top Memory Users Sample Output

2.5 TOPAS OR TOP SYSTEM MONITORING

Topas command is a performance monitoring tool that is ideal for performance analysis. Its name is varying from operating system version. It is capable of reporting local system statistics such as CPU usage, queues, memory and paging use, disk performance, and network performance. All information is real time.



```
Topas Monitor for host:    ay                   EVENTS/QUEUES    FILE/TTY
Sun Jul 27 21:48:38 2008   Interval:  2         Cswitch    3057  Readch     0.3G
                                                Syscall  682.0K  Writech 2016.7K
Kernel   19.3   |########                    |  Reads     88845  Rawin         0
User     80.7   |########################    |  Writes     1401  Ttyout      762
Wait      0.0   |                            |  Forks       198  Igets         0
Idle      0.0   |                            |  Execs       206  Namei     16955
Physc =  4.00                    %Entc= 100.0   Runqueue   28.0  Dirblk        0
                                                Waitqueue   0.0
Network  KBPS   I-Pack  O-Pack   KB-In  KB-Out
en0     908.9   429.0   355.5   106.3   802.6   PAGING           MEMORY
lo0      42.5    27.5    28.0    21.3    21.3   Faults    49164  Real,MB    7744
                                                Steals        0  % Comp     69.9
Disk    Busy%     KBPS     TPS KB-Read KB-Writ  PgspIn        0  % Noncomp  28.8
hdisk2    4.0    74.0    18.5     0.0    74.0   PgspOut       0  % Client   28.8
hdisk3    3.5    74.0    18.5     0.0    74.0   PageIn      279
hdisk155  1.5     4.0     0.5     4.0     0.0   PageOut       3  PAGING SPACE
hdisk417  1.0   152.0     2.5   152.0     0.0   Sios        282  Size,MB    8192
hdisk420  1.0   218.0     3.0   218.0     0.0                    % Used      0.0
                                                NFS (calls/sec)  % Free    100.0
Name            PID  CPU%  PgSp Owner           ServerV2       0
rrdtool     6615286   1.9   1.8 admbot          ClientV2       0   Press:
rrdtool     6226130   1.7   1.4 admbot          ServerV3       0   "h" for help
rrdtool     5296354   0.8   0.6 admbot          ClientV3       0   "q" to quit
perl        7417866   0.6   1.7 admbot
perl        6480118   0.5   1.7 admbot
```

Figure 2.11 : Topas Sample Output

As seen from Figure 2.11 the important components are as fallows;

CPU utilization CPU utilization is graphically and numerically displayed below the date and time. Values are same as SAR or Vmstat output.

Network statistics	Network throughput for each network adapter can be seen with this column. All input or output packages as KBps.
Disk statistics	From the Disk part a limited number of disks can be watched. Not every disk activity. But it is sorted by percentage of usage.
Process statistics	The top CPU user processes are displayed with process id, CPU usage percentage and process owner.
Other Statics	Other shown statics are; total run queue, File reads and Writes, Paging and memory static, Nfs requests.

Topas has a lot of advantages but it also has several disadvantages;

i. Topas is a great tool to monitor system performance. But in order to run Topas user must login to system. Another disadvantage is, it is not possible to monitor several systems from the same window.

ii. It has no output file. Output can’t be saved for future analyses. So workload increases over time can’t be gathered.

iii. More than one user can run Topas. Each of these threats needs CPU power to run. So it can be very expensive for system resources.

3. WINDOWS PERFORMANCE MONITORING

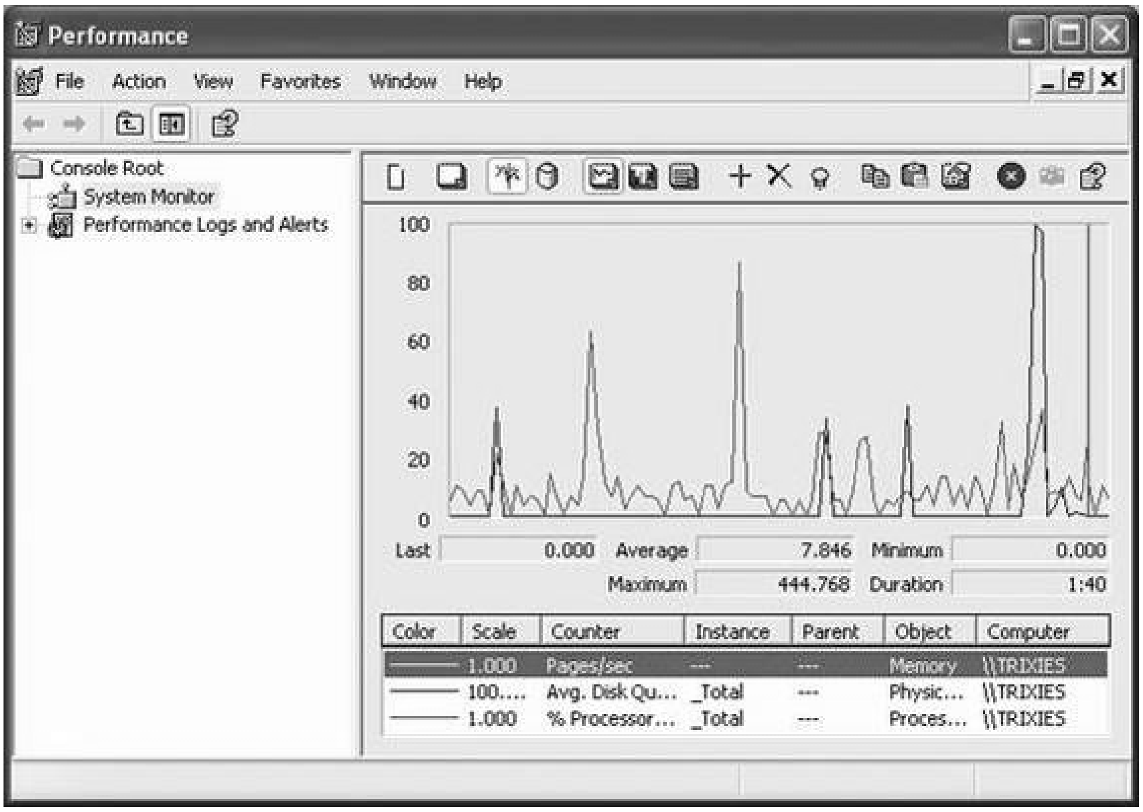
As mission critical applications mostly running on Windows servers, it also important to monitor performance bottlenecks. As Microsoft's point of view, they developed Microsoft Operations Framework (MOF) which monitors servers for operating system errors and warnings, also monitors base CPU monitoring but nothing more.

Windows servers performance monitoring can be done by build-in performance counters. There is also a tool called "Performance Monitor" which gives performance information when it's is configured as needed. This tool uses build in performance counters too. The Windows Performance Monitor is a good diagnostic tool to compare the collected data and keep it as a record for problem analysis. From this single monitoring console a range of system processes can be tracked and real time graphical display of results will be reported.

Performance Monitor application has a series of three different views on system performance (homenetworkhelp.info, 2009):

i. Chart view - This is the primary default view which allows objects to be graphically displayed. This view enables you to view the monitored items over a short period of time (as short as every second) by choosing the options from within the dialog boxes.
ii. Alert view - This view will enable you to do background monitoring of the system while working with other applications.
iii. Counter and Trace log - These log views enables you to record the selected counters into a log file. The log file would be examined later to find potential and existing bottlenecks.

In Figure 3.1 CPU usage, disk usage and swap usage data can be seen. Although it is possible to monitor a huge number of components, the graphs are a bit confused.


Performance
File Action View Favorites Window Help
Console Root
System Monitor
Performance Logs and Alerts
100
80
60
40
20
0

Last	0.000	Average	7.846	Minimum	0.000
		Maximum	444.768	Duration	1:40

Color	Scale	Counter	Instance	Parent	Object	Computer
——	1.000	Pages/sec	---	---	Memory	\\TRIXIES
——	100....	Avg. Disk Qu...	_Total	---	Physic...	\\TRIXIES
——	1.000	% Processor...	_Total	---	Proces...	\\TRIXIES

Figure 3.1 : Windows Performance Monitor Sample

Some of the counters are;

- Memory
  - Memory Pages/Sec
  - Paging file percent Usage
- CPU
  - Percent Processor Time
- Disks
  - Disk Queue Length
  - Average disk Sec/Transfer
  - Percent disk time

Altough performance monitor in windows is usefull, it is not possible to monitor several servers in same window. in Iris monitoring system the same performance counters are collected to obtain bottlenecks.

4. DEVELOPMENT OF A MONITORING SYSTEM (IRIS)

This chapter describes the development phrases of a system performance monitoring tool called Iris. This tool has been developed to assist performance analysis and monitoring in UNIX and Windows systems.

4.1 REQUIREMENTS ANALYSES

It is first decided to inspect commercial software that monitors every server with no effect on server. Since there was no product that would fulfill the entire requirement it is decided to examine built in solutions which are; Nmon output for Aix, SAR and Vmstat output for Solaris, build-in performance counters for Windows. But these tools have no GUI and even graphical entities such as charts were produced via the command line interface. As a result, the ability to correlate data from different performance commands was absent.

The major requirements are; no client application must be installed on clients, all data are must be collected and processed online, graphs and other useful information must be conveyed to users.

4.2 THE DESIGN PHASE

The first part of design phases is designing security infrastructure, because everything will be built on that principal. The most important aspect of security design was to design a product with minimal security concerns. Since this product would be connecting to all UNIX and Windows servers, a high level of attention to security was required.

After several iterations, system designed with a security model that uses non-authorities user for UNIX environment, a special connection port for servers on DMZ, and unfortunately a Domain Admin user for Windows environment. Because performance counters are only available to power users and because of our design the same user must be connecting to several servers.

Another design issue is applications portability capability. Selected Programming language for this design is Korn Shell for UNIX servers. Because Korn Shell is included in nearly all UNIX versions by default. Other choices like Java or C is not preferred because, Java is only included in SUN Solaris by default so it needs additional installations in different UNIX servers and also C needs to be installed and compiled for each server because of library files can vary on servers. For Windows performance monitoring Virtual Basic Scripting is selected for its simple and powerful architecture. For Graph pages CGI and Html was used with Apache web server.

The last part of design is database choice for storing data. Round Robin Database is used for data stores because of its abilities for storing and graphing data.

Now detailed information about used technologies will be given;

4.2.1 Korn Shell

First of all what is Shell. A UNIX shell, is a command interpreter and script host that provides a traditional user interface for the UNIX operating system and for UNIX-like systems. It is also called a "shell" because it hides the details of the underlying operating system behind the shell's interface (Wikipedia, 2009)

The Korn shell (Ksh) is a UNIX shell which was developed by David Korn in 1980s. The Korn Shell language is also a complete, powerful, high-level programming language for writing applications, often more easily and quickly than with other high-level languages. (www.kornshell.com,2009) Before Ksh there are two popular shells which are Bourne Shell (Sh) and C Shell (Csh). Ksh has the best features of these both shells plus programming support. The new version of Ksh also has the functionality of other scripting languages such as Awk or Perl

Like Visual Basic Scripting, Ksh is integrated in all UNIX versions (it is also default shell in IBM AIX). Also it is very easy to use performance management tools like SAR or Vmstat in Ksh scripts.

4.2.2 Virtual Basic Scripting

VBScript is an interpreted script language from Microsoft that is a subset of its Visual Basic programming language designed for interpretation by Web browsers (SearchEnterpriseDesktop, 2009). VBScript began as part of the Microsoft Windows Script Technologies, which were targeted at web developers initially and were launched in 1996. (Wikipedia, 2009)

VBScript is installed by default in every desktop release of Microsoft Windows since Windows 98. It initially gained support from Windows administrators seeking an automation tool more powerful than the batch language first developed in the late 1970s (Wikipedia, 2009). By the end of 2008, no new functionality will be added to the VBScript language, which has been superseded by Windows Power Shell. However, it will continue to be shipped with future releases of Microsoft Windows. This is the major point that it is selected for windows environment. It is already bundled with servers so there is no need to install anything on servers.

One of the advantages of VBScript (in common with other scripting languages) is that it's written in plain, ordinary ASCII text. That means that development environment can be something as simple as Notepad.

4.2.3 CGI and Html

The Common Gateway Interface (CGI) is a standard for interfacing external applications with information servers, such as HTTP or Web servers (hoohoo.ncsa.uiuc.edu, 2009).

Instead of an Html pages, a small program’s or script’s output is displayed on the browser with the help of CGI. Html pages does not change and are called static. However, a CGI script is an executable program and output is dynamic.

A CGI program can be written in any language that allows it to be executed on the system, such as: (hoohoo.ncsa.uiuc.edu, 2009)

- C/C++
- Fortran
- PERL
- TCL
- Any UNIX shell
- Visual Basic
- AppleScript

Perl language is preferred because it is also installed on main server. Why Cgi is used? Because when performance data’s comes from clients additional UNIX scripts must be called with in server to update database and update graphic files. With the help of Cgi scripts other UNIX Ksh scripts can easily started and results can be watched.

4.2.4 Apache Web Server

The Apache HTTP Server Project is a collaborative software development effort aimed at creating a robust, commercial-grade, featureful and freely-available source code implementation of an HTTP (Web) server (apache,2009). The first version of the Apache web server, based on NCSA httpd web server was created by Robert McCool at University of Illinois. Later Core development of the Apache Web server is performed by a group of about 20 volunteer programmers, called the Apache Group.

Apache is primarily used to serve both static content and dynamic Web pages on the World Wide Web. Many web applications are designed expecting the environment and features that Apache provides.

The original version of Apache was written for UNIX, but there are now versions that run under OS/2, Windows and other platforms. Since April 1996 Apache has been the most popular HTTP server on the World Wide Web. As of December 2008 Apache served over 51 percent of all websites. (Wikipedia, 2009)

Since Apache is an open-source everyone can install, use or add additional features. The main purpose of selecting apache is; its working stable on UNIX servers and the project is mainly build on that platform. With the help of Apache http and CGI pages are serving.

4.2.5 RRDtool

RRDtool (Round Robin Database tool) is a system to store and display time-series data. it is It stores the data in a very compact way. it has a fixed amount of size and data's are written in round-robin algoritm. the main architecture of RRD is very simple. like standart databases the primary key is the time which data will be inserted. after all the data's inserted, graphical views can be created within rrdtool (wikipedia, 2009).

Every performance data and graphs are created and stored with the help of this tool. this tool is developped by Tobi Oetiker and freely distributed on GNU General Public Licence.

4.3 DEVELOPMENT PHASE

In this section, development phrases of Performance Monitoring application is introduced. Information will be given in two parts, First UNIX Performance monitoring and second Windows Performance monitoring according to Figure 4.1.

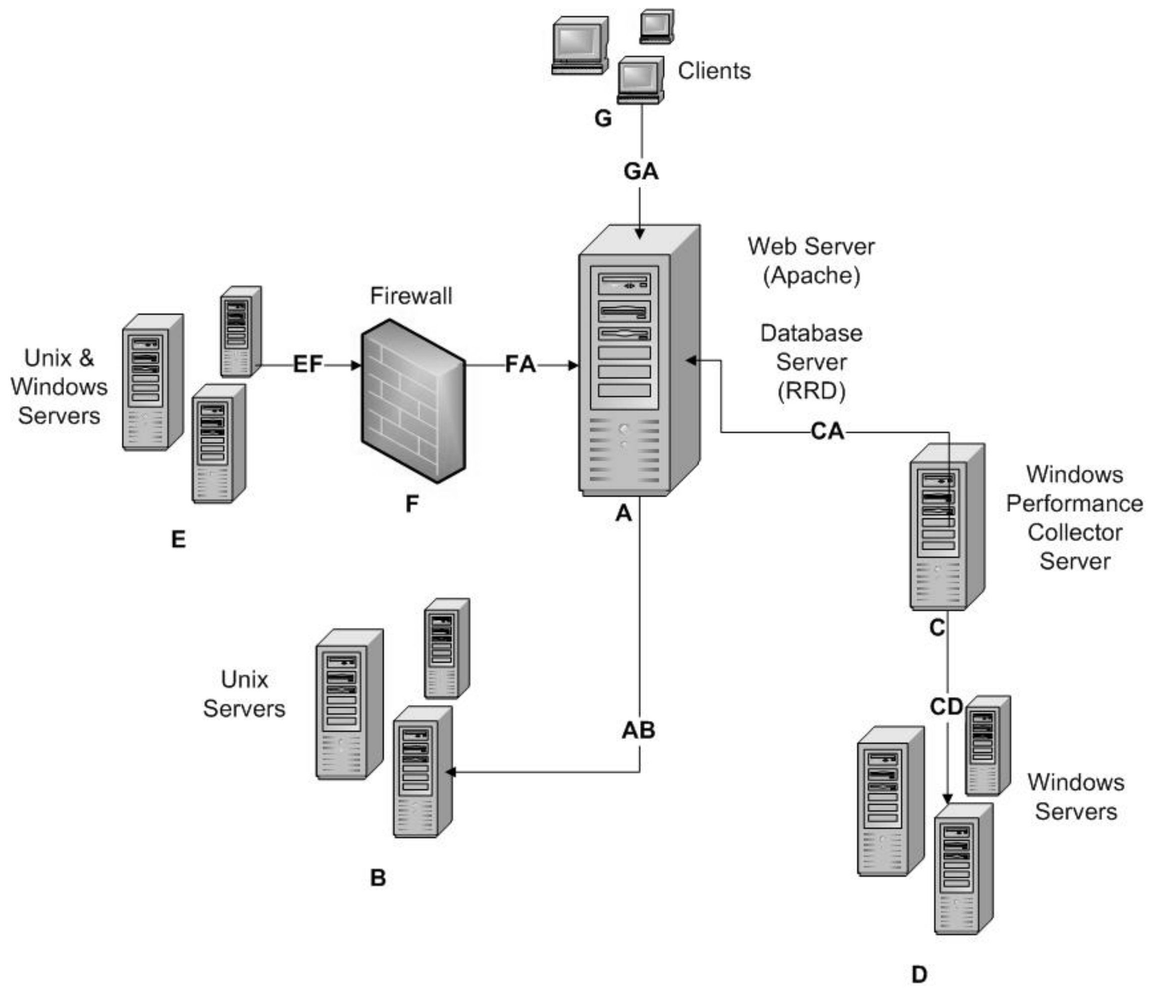


Figure 4.1 : Iris Design Schema

In Figure 4.1 the main architecture consists of one DB server and one Windows Performance Collector server. Main DB server is serving Html pages and all of the RRD files stands on this server. The major advantage of this architecture is its simple design. More servers can be added to monitoring structure. It only changes the B, D and E sites (in Figure 4.1), other elements remains same. This design is also totally different from designs of previous works. Because neither of previous works can collect both Windows and UNIX performance. They mostly focus on one type and also one version of operating systems.

4.3.1 UNIX Performance Monitoring Development

4.3.1.1 Round robin database file creation

Development of UNIX performance monitoring system begins with Database design and creation. The Database (DB) server is shown at Figure 4.1 with label "A". As in RRD design chapter the main design of RRD DB is very important. Because it can store a fixed amount of data and it can't be changed.

Agreed amount of time the data's will be stored is 5 years. Figure 3.2 shows the main design of CPU and memory DB files.

```
/opt/freeware/bin/rrdtool create $path/$name.rrd --start $start --step 60 \
DS:id:GAUGE:90:0:100 \
DS:wait:GAUGE:90:0:100 \
DS:usr:GAUGE:90:0:100 \
DS:sys:GAUGE:90:0:100 \
DS:memUsed:GAUGE:90:0:100 \
DS:memFree:GAUGE:90:0:100 \
DS:cpuTotal:GAUGE:90:0:1000 \
RRA:MIN:0.60:1:2628000 \
RRA:MAX:0.60:1:2628000 \
RRA:AVERAGE:0.60:1:2628000
```

Figure 4.2 : Cpu and Memory RRD Database

The variables in Figure 4.2 are;
$path variable points the place where DB files will be stored.
$name variable is the name of DB file, this name will always be the name of server.
$start value indicates when the DB file is created.

With DS parameter columns for DB file are defined. The main columns are id (CPU idle time), wait (CPU I/O wait time), usr (CPU usr time), sys (CPU Kernel time), memUsed (Total used memory in system), MemFree (Free memory in system), cpuTotal (Total number of CPU in system).

Each column will be update in sixty seconds. And there will be 2628000 records which makes exactly 5 years record. (1440 records for 1 day, 525600 records for 365 days) Figure 4.2 is an example of DB file creation. Any type of DB file can be created for monitoring any type of numeral variable. For example For Core Banking Server; is also important to monitor banking application online session numbers, number of Core Banking operations that will successfully ends and Oracle performance parameters. Like these server-special exceptions, the system is totally adoptable.

4.3.1.2 Performance data collection

As in introduced in Chapter 4.2 of thesis, KSH scripting will be used for performance data collection. Outputs of UNIX performance tools like SAR, Vmstat, Iostat and Ps are used to show the performance status of servers.

First SSH is used for communication between servers with an unprivileged user (AB labeled way in Figure 4.1). For this purpose a SSH key is generated with "ssh-keygen –t dsa" command at the main server. After SSH public and private key pair will be created, public key will be used for connecting to remote clients.

The main KSH script uses the outputs of UNIX tools described above and updates the databases files. Figure 4.3 shows and example part of KSH scripts that collect information (only some lines).

```
/home/admbot/script/create_database/rrdCheck.ksh $i "unix"
proceess=`ssh $i ps -ef|wc -l| tr -s " "| sed 's/^[ ]//g'`
datam=`ssh $i /usr/sbin/sar 1 1|tail -1`
deger=`ssh $i /usr/bin/svmon |head -3|tail -2`
cpu_total=`ssh $i lsdev -Cc processor|grep Available|wc -l| tr -s " "| sed 's/^[ ]//g'`
data=`ssh $i /usr/bin/iostat -s|head -9|tail -1`
```

Figure 4.3 : UNIX Performance Data Collection Example

As seen from Figure 4.3, First script check if the database file exists. If no database file exists than it first creates the RRD file. Then is gathers Ps, SAR, Vmstat, Svmon (used for total size of memory), number of CPU, and I/O information from server. Then script makes some decomposition on values and it updates the database at the end. The whole processed are shown in Figure 4.4.

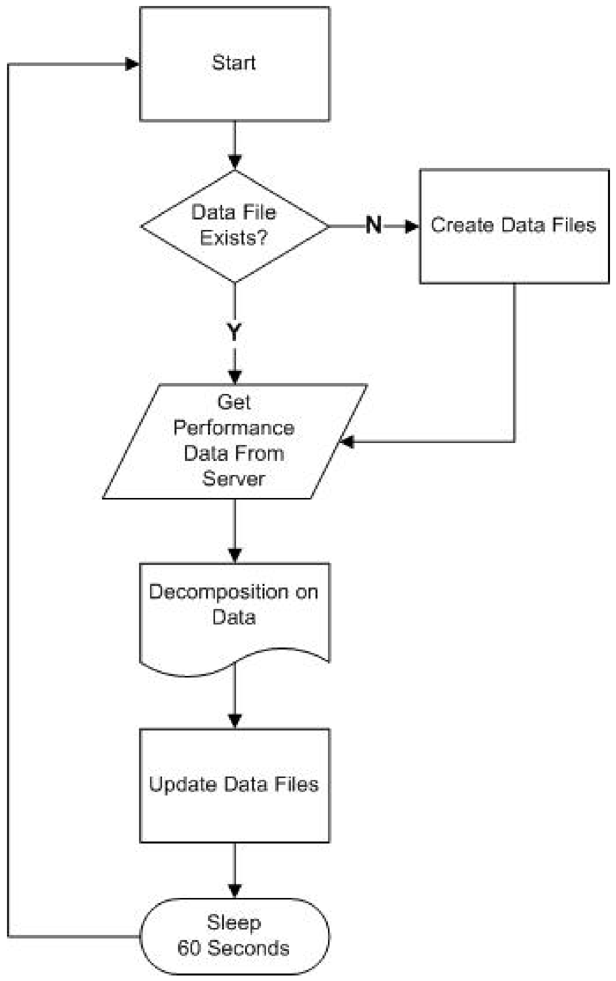
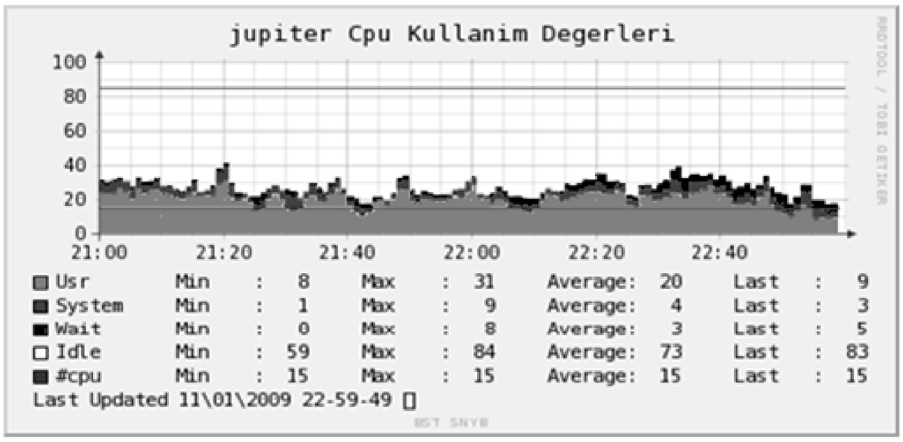


Figure 4.4 : UNIX Performance Collection Flow Chart

Like CPU performance monitoring, Memory and paging activity is gathered with Vmstat, I/O activity is gathered with Iostat and decomposed with algorithms and updates the memory database file.

4.3.1.3 Sample graphs

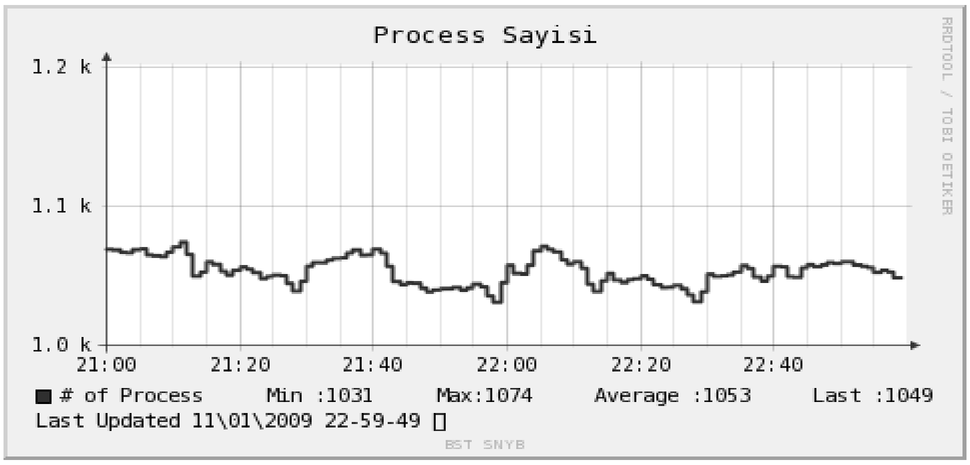
In this part sample outputs of above work will be given. Figure 4.5 is the example of CPU usage monitoring. The values are gathered from SAR command. This graph also shows the number of processor in system.



```
▨ Usr      Min  :  8   Max  :  31   Average:  20   Last  :  9
■ System   Min  :  1   Max  :   9   Average:   4   Last  :  3
■ Wait     Min  :  0   Max  :   8   Average:   3   Last  :  5
□ Idle     Min  :  59  Max  :  84   Average:  73   Last  :  83
■ #cpu     Min  :  15  Max  :  15   Average:  15   Last  :  15
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.5 : Iris CPU Usage Graph Example (SAR)

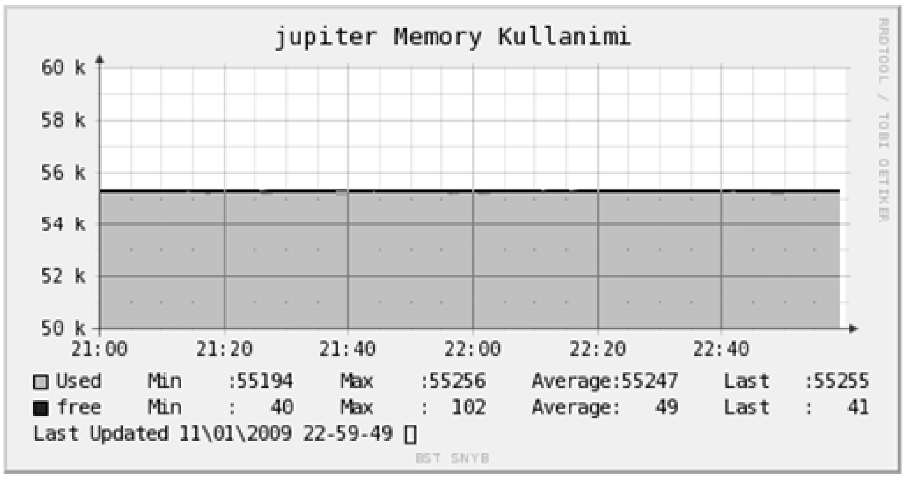
Figure 4.6 show the number of total processes in system. It is also important to watch this value for abnormal process activity. The values are gathered from Ps command.



```
■ # of Process     Min :1031     Max:1074     Average :1053     Last :1049
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.6 : Iris Process Count Graph (PS)

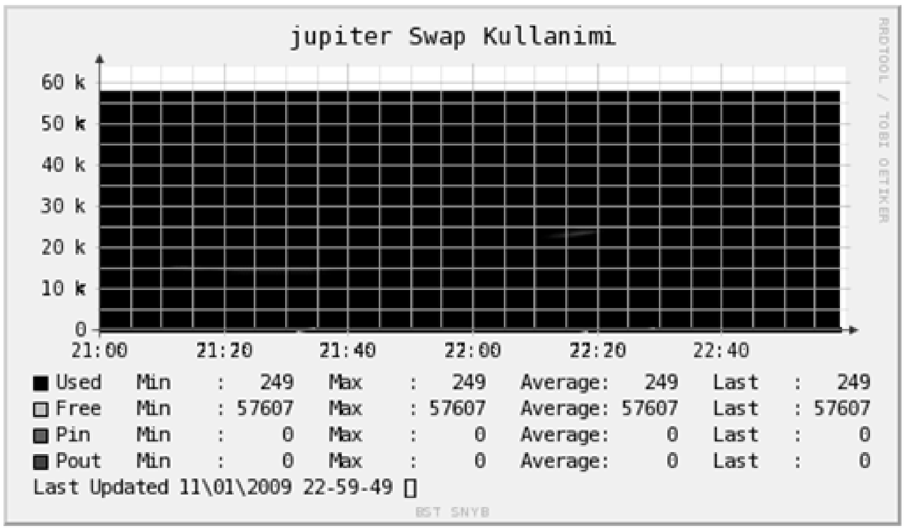
Figure 4.7 shows used and free memory in system. As seen in figure there is only 40Mb of free memory. Although it is normal on current system setting, watching available memory is a mission critical.



```
□ Used   Min   :55194   Max   :55256   Average:55247   Last   :55255
■ free   Min   :   40   Max   :  102   Average:   49   Last   :   41
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.7 : Iris Memory Usage Graph (Vmstat)

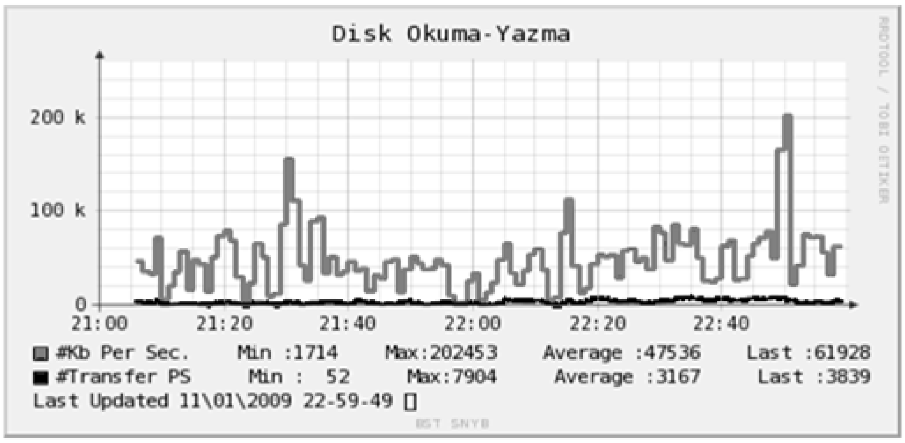
It is also important to watch available and used swap space in system as seen from Figure 4.8.



```
■ Used   Min   :   249   Max   :   249   Average:   249   Last   :   249
□ Free   Min   : 57607   Max   : 57607   Average: 57607   Last   : 57607
■ Pin    Min   :     0   Max   :     0   Average:     0   Last   :     0
■ Pout   Min   :     0   Max   :     0   Average:     0   Last   :     0
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.8 : Iris Swap Usage Graph (Vmstat)

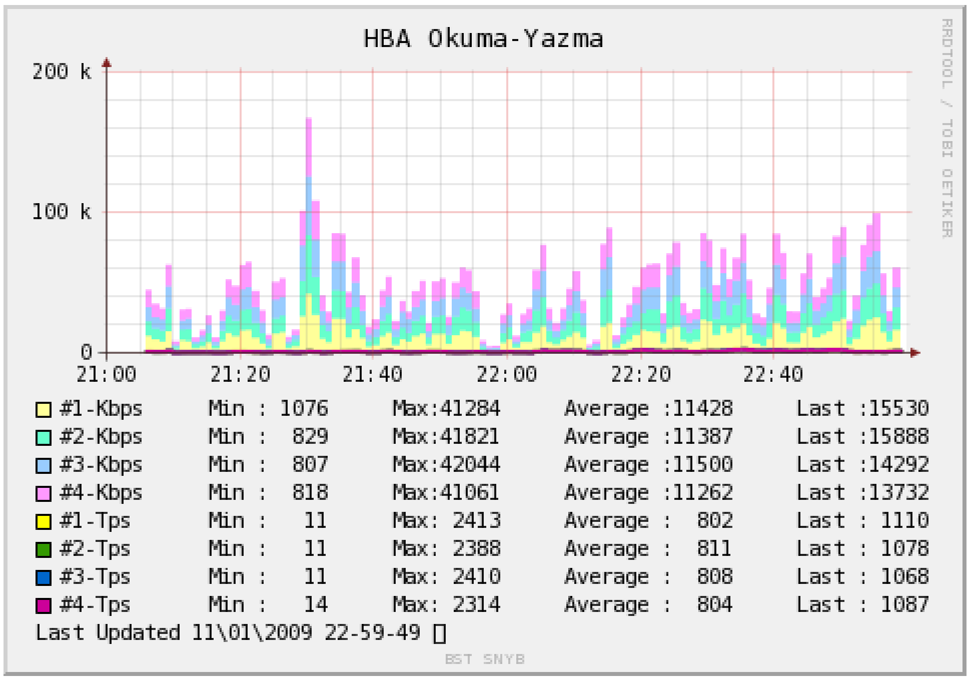
In Figure 4.9 the total I/O activity in system is shown. In a bigger time scale it is important to have I/O series in similar. This graph also contains I/O to local attached disks and tape drives.



```
■ #Kb Per Sec.     Min :1714    Max:202453    Average :47536    Last :61928
■ #Transfer PS     Min :  52    Max:7904      Average :3167     Last :3839
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.9 : Iris Total Disk I/O Graph (Iostat)

For the fiber connected SAN disks, the total throughput can be seen in Figure 4.10. The most important thing to consider is the amount of data passes though each adapter must be very close.



```
□ #1-Kbps    Min : 1076    Max:41284    Average :11428    Last :15530
□ #2-Kbps    Min :  829    Max:41821    Average :11387    Last :15888
□ #3-Kbps    Min :  807    Max:42044    Average :11500    Last :14292
□ #4-Kbps    Min :  818    Max:41061    Average :11262    Last :13732
□ #1-Tps     Min :   11    Max: 2413    Average :   802   Last : 1110
■ #2-Tps     Min :   11    Max: 2388    Average :   811   Last : 1078
■ #3-Tps     Min :   11    Max: 2410    Average :   808   Last : 1068
■ #4-Tps     Min :   14    Max: 2314    Average :   804   Last : 1087
Last Updated 11\01\2009 22-59-49 []
```

Figure 4.10 : Iris Fiber HBA Graph (Iostat)

As a result major UNIX tools are used to build a real-time monitoring system. It is also very simple but very useful.

4.3.2 Windows Performance Monitoring

In Windows performance monitoring, build in Windows Performance Classes (WPC) were used to collect data. These classes are predefined in Windows servers with sub functions which returns performance related information. Detailed information about classes that were used will be given in this chapter.

The main architecture is a bit different than UNIX part, because a central Windows server is needed to access to windows clients (Server C in Figure 4.1). Window servers can be accessed from UNIX but user permissions can be problem. So a central server is created and Visual Basic scripts are working on this main server. In this architecture main server is collecting the client performance with a domain admin user (Path CD in Figure 4.1), then it sends the data to main UNIX server (Path CA in Figure 4.1).

The most critical parameters of windows performance monitoring doesn't differ from UNIX. CPU, memory, swap, disk and network parameters are going to be monitored. The fresh data is gathered from WPC which are;

- Win32_PerfFormattedData_PerfOS_Processor
- Win32_PerfFormattedData_PerfOS_Memory
- Win32_PerfFormattedData_PerfDisk_PhysicalDisk
- Win32_PerfFormattedData_TCPIP_NetworkInterface

Now more information about these classes will be given;

Win32_PerfFormattedData_PerfOS_Processor

This performance counter class provides pre-calculated data from performance counters that monitor aspects of processor activity (msdn.microsoft.com, 2009). Like SAR in UNIX, how much CPU is in use or other CPU related information can be gathered with this performance class.

The algorithm is very simple. For each processor running in system PercentProcessorTime function is called, which is predefined in

Win32_PerfFormattedData_PerfOS_Processor class, and this function returns a value that gives the load of this CPU. After all information's from all CPU's are gathered the load on main server is gathered by dividing the load to number of CPU.

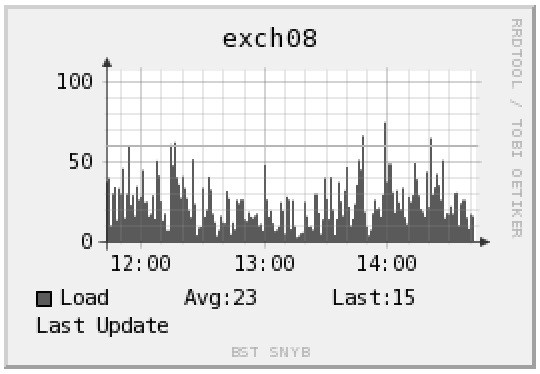


Figure 4.11 : Windows CPU Usage Sample

In Figure 4.11, CPU usage of an Exchange server can be seen. This server has four CPU and the main load of server is gathered by calculating loads on each CPU. (Windows performance graphs are smaller than UNIX graphs because the number of windows servers is much more than UNIX servers.)

Win32_PerfFormattedData_PerfOS_Memory

The Memory class provides pre-calculated performance data from performance counters that monitor the physical and virtual memory on the computer. Physical memory is the amount of random access memory (RAM) on the computer and Virtual memory consists of space in physical memory and on disk (msdn.microsoft.com, 2009)

This class gives every information about system memory's. The important values for monitoring are; total and free physical memory, total and usage virtual memory.

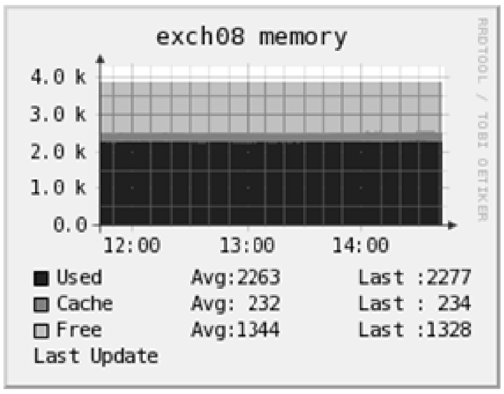


Figure 4.12 : Windows Physical Memory Sample

In Figure 4.12, monitoring of physical memory can be seen. With the help of this tool free or used memory can easily be seen (values are in MB). Free memory value is gathered from AvailableMBytes function, cache value is gathered from CacheBytes function. Total memory is gathered from Win32_ComputerSystem class which contains general information about running system (this class values can be seen from Control Panel -> system in any windows).

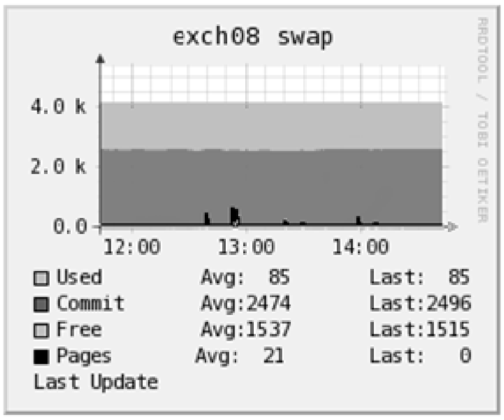


Figure 4.13 : Windows Virtual Memory Sample

In Figure 4.13 most important of virtual memory statics can be seen. Used value show the amount of virtual memory in system. Committed memory is reserved space in swap space for the running processes. Amount of this commit memory varies on the amount of physical memory that process uses. So it is important to watch available and committed memory of a windows system.

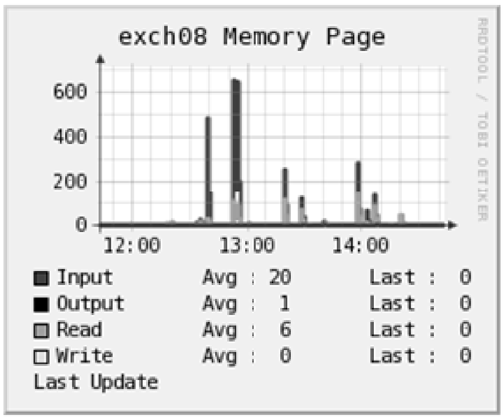


Figure 4.14 : Windows Virtual Memory Operations Sample

In Figure 4.14 operations on virtual memory can be seen. These operations contains page- in and page-out activities. Input and output values show the total number of pages which are written or read from virtual memory. Read and Write variables show the numbers of times that read or write requests comes.

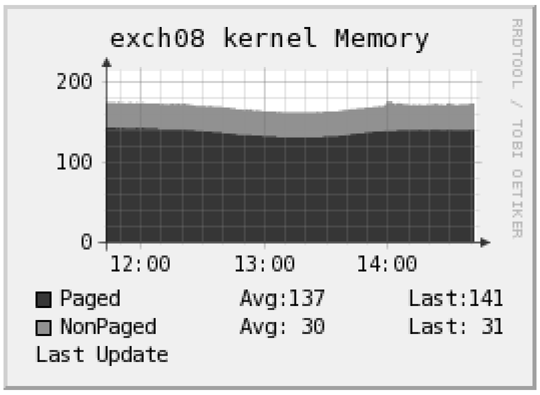


Figure 4.15 : Windows Kernel Memory Sample

In Figure 4.15 the amount of paged and non paged memory can be seen. NonPaged pool is the operating system memory region in physical memory that can't be written to virtual memory. Paged pool is also operating system memory region which can be written to disk.

Win32_PerfFormattedData_PerfDisk_PhysicalDisk

Physical Disk formatted data class provides pre-calculated data from performance counters that monitor hard or fixed disk drives on a computer (msdn.microsoft.com, 2009). With disk monitoring in Windows only local attached disks will be considered.

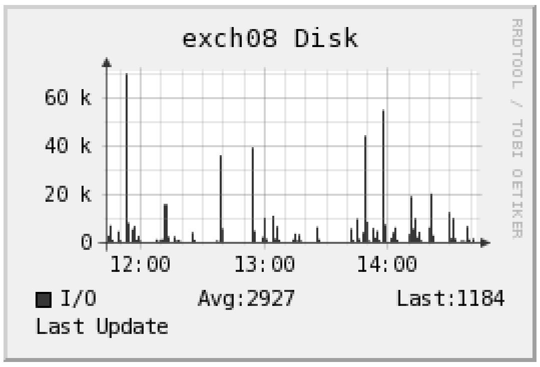


Figure 4.16 : Windows Disk I/O Sample

As seen from Figure 4.16 I/O's to local disks are gathered from DiskBytesPerSec function. Although this function gives information about all disks, total value of all disks will be graphed for a simple view. The values are in MB.

Win32_PerfFormattedData_TCPIP_NetworkInterface

Network interface class provides pre-calculated data from performance counters that monitor the rates at which bytes and packets are sent and received over a TCP/IP network connection (msdn.microsoft.com, 2009).

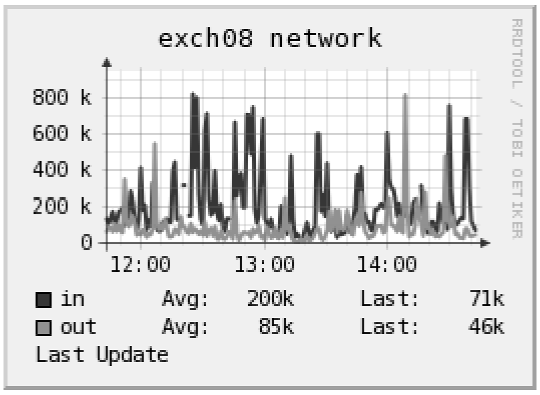


Figure 4.17 : Windows Network I/O Sample

Figure 4.17 is the sample of network monitoring. The monitoring variables are input and output values. These values are gathered by PacketsReceivedPerSec and PacketsSendPerSec functions.

As a result windows performance management can be done by getting values from build-in performance counters. The main monitoring structure is so adoptable that every parameter that needs to be monitored can be added as a sub part.

5. RESULT AND DISCUSSION; ORACLE REDO LOGS PERFORMANCE PROBLEM

As the Iris monitoring system is deployed to all mission critical UNIX and Windows servers, they have been monitored in real time. The most important benefit of this work is Server Operation teams are watching graphs 7/24 for performance problems.

In this part a problem in Core Banking system will be examined and with the help of Iris monitoring, effects of tunings made to server will be shown. Core banking server is the main DB server of all applications. Whole customer information and other banking related information is stored in this DB. Every money related operations made in branches commits to this DB.

5.1 DESCRIPTION OF THE PROBLEM

The major problem occurs when management decides to build a more functional Disaster Recovery (DR) site and in this new design of project Oracle Data Guard is selected for database (DB) replications.

Oracle Data Guard is an extension to the Oracle RDBMS. It aids in establishing and maintaining secondary "standby databases" as alternative/supplementary repositories to production "primary databases" (Wikipedia, 2009). Data Guard maintains standby databases as consistent copies of production databases. And in a case of failure in production site, Data Guard can switch these standby DB's to the production DB's role. This architecture acts as a cluster like DB. Because of this design, all of the production data's must be read and written to DR site. This makes a second read load on servers and disks.

The working mechanism of Oracle Data Guard is based on redo logs. When oracle gets an update or insert command it first write this command to redo logs. After user commits the command another data is written to redo log. So all the operations on database will be held in redo logs. Data guard automatically copies this redo log to disaster recovery and applies it to standby server. These operations occur simultaneously. All of these operations can be seen in Figure 5.1.

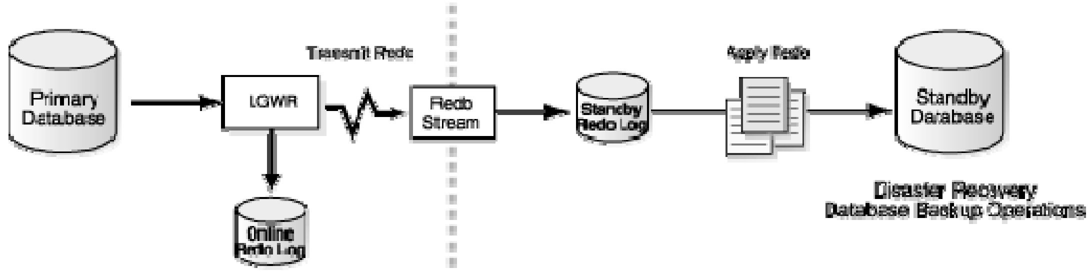
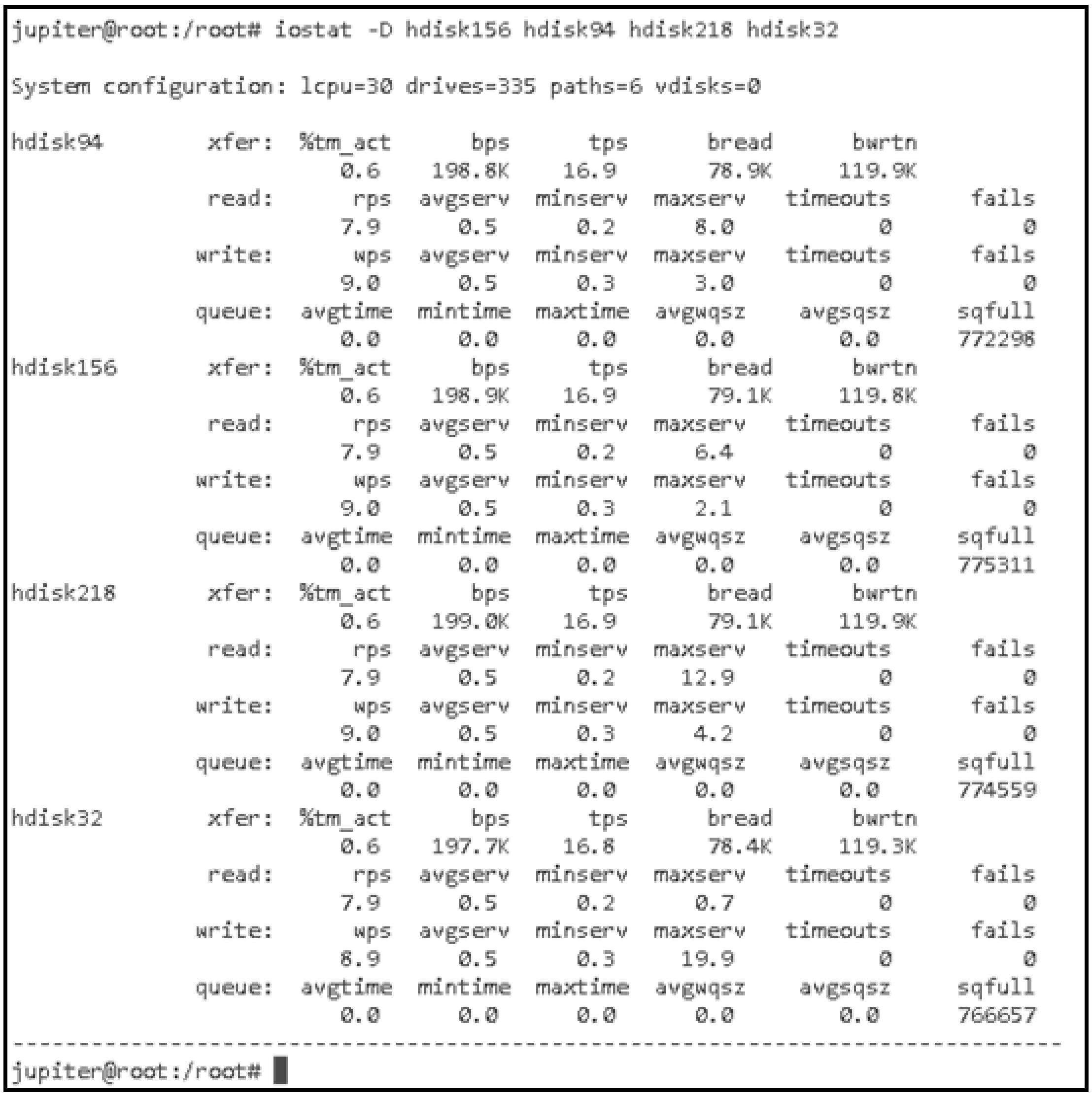


Figure 5.1 : Oracle Data Guard (Oracle.com, 2009)

With these Redo log and Disaster Recovery operations, the first problem occurs. Because of this increased I/O, disks of redo logs starts getting service full warnings. The Service Full (Sqfull) means that; working disk device queue is not enough for requests to write data to disk. Some of the write requests didn't even enter the storage system's queue, they remain in operating systems write queue. This problem is so critical that every banking operation that customers make in branches gets higher waiting time or even gets timeout errors. In this situation several performance tuning options are considered. These considerations will examined and discussed in later sections.

5.2 IOSTAT AND SQFULL VALUES

As introduced in the "UNIX Performance Monitoring" part (Chapter 2.3) Iostat is used to monitor disk activity. It also generates reports that can be used to change system configuration to better balance the load between physical disks and fiber channel adapters. Iostat command is also useful to determine whether a physical volume is becoming a performance bottleneck and if there is potential chance to improve the situation as in our situation.



```
jupiter@root:/root# iostat -D hdisk156 hdisk94 hdisk218 hdisk32

System configuration: lcpu=30 drives=335 paths=6 vdisks=0

hdisk94        xfer:  %tm_act      bps      tps      bread      bwrtn
                        0.6      198.8K     16.9      78.9K     119.9K
               read:      rps   avgserv  minserv  maxserv   timeouts      fails
                        7.9       0.5      0.2      8.0          0          0
              write:      wps   avgserv  minserv  maxserv   timeouts      fails
                        9.0       0.5      0.3      3.0          0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0       0.0      0.0      0.0        0.0     772298
hdisk156       xfer:  %tm_act      bps      tps      bread      bwrtn
                        0.6      198.9K     16.9      79.1K     119.8K
               read:      rps   avgserv  minserv  maxserv   timeouts      fails
                        7.9       0.5      0.2      6.4          0          0
              write:      wps   avgserv  minserv  maxserv   timeouts      fails
                        9.0       0.5      0.3      2.1          0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0       0.0      0.0      0.0        0.0     775311
hdisk218       xfer:  %tm_act      bps      tps      bread      bwrtn
                        0.6      199.0K     16.9      79.1K     119.9K
               read:      rps   avgserv  minserv  maxserv   timeouts      fails
                        7.9       0.5      0.2     12.9          0          0
              write:      wps   avgserv  minserv  maxserv   timeouts      fails
                        9.0       0.5      0.3      4.2          0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0       0.0      0.0      0.0        0.0     774559
hdisk32        xfer:  %tm_act      bps      tps      bread      bwrtn
                        0.6      197.7K     16.8      78.4K     119.3K
               read:      rps   avgserv  minserv  maxserv   timeouts      fails
                        7.9       0.5      0.2      0.7          0          0
              write:      wps   avgserv  minserv  maxserv   timeouts      fails
                        8.9       0.5      0.3     19.9          0          0
              queue:  avgtime  mintime  maxtime  avgwqsz    avgsqsz     sqfull
                        0.0       0.0      0.0      0.0        0.0     766657
--------------------------------------------------------------------------------
jupiter@root:/root# █
```

Figure 5.2 : Disk health with Iostat

Figure 5.2 is the sample output of Iostat command for redo log’s disks. As seen in Figure 5.2, hdisk32, hdisk94, hdisk156 and hdisk218 are using 0.6 percent and getting nearly 200Kbps. This is usually below the normal utilization, but Sqfull value is enormously high. They have average 774,000 times. Which means when Oracle redo log’s write or read threats needs to access disk, it gets service is full error more than 774,000 times. These rejected disk access requests results in wait I/O in CPU and timeouts or longer waiting times for customers.

5.3 PARAMETERS AFFECTING PERFORMANCE IMPROVEMENT

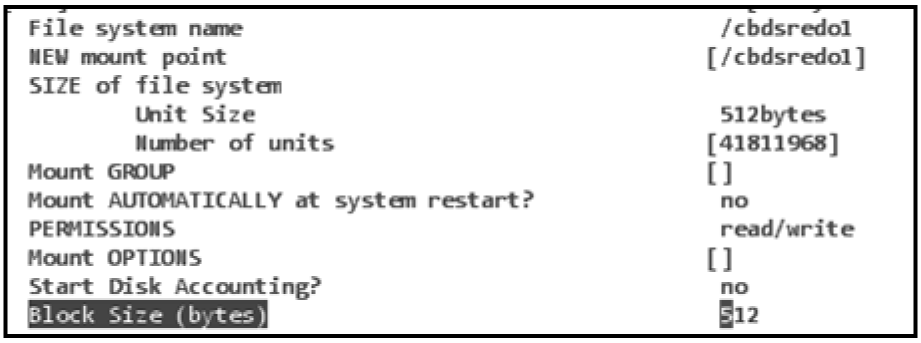
In this part of thesis several performance improvement scenarios are going to be discussed. After consulting them for their practicable behavior, Iris system is used to observe performance improvements on system.

5.3.1 Changing the Storage Structure of Redo Log Disks

Changing the storage structure means the disk structure of file systems (FS). In general Oracle structure, there will be 2 identical copies of each redo logs (This is also a best practice advice from Oracle). In production system, these file systems were created on 5 GB disks on IBM high-end storage system.

The first option is striping the file systems. Striping means adding two different disks together. I/O requests are divided between these stripes file systems equally. Striping is generally used when file system needs to be bigger and highly performance. Because the redo FS's are too small striping won't make any performance improvements.

File System Block size is another option for tuning. Because when the file systems buffer cache will be bypassed (with direct I/O or concurrent I/O), performance depends on the file system block size (agblksize). For optimal performance, agblksize must be equal to the size of the smallest block accessing the file system, for the redo log files, the smallest block size used is 512 bytes.



```
File system name                                      /cbdsredo1
NEW mount point                                      [/cbdsredo1]
SIZE of file system
        Unit Size                                     512bytes
        Number of units                              [41811968]
Mount GROUP                                          []
Mount AUTOMATICALLY at system restart?                no
PERMISSIONS                                           read/write
Mount OPTIONS                                        []
Start Disk Accounting?                                no
Block Size (bytes)                                    512
```

Figure 5.3 : File system Block Size Output

As in Figure 5.3, Block size value was set to optimum value. Also IBM recommends setting database agblksize size to 4096 where database block size is bigger than 2048bytes. Other file systems block size is recommended to set to 512 bytes.

5.3.2 Increasing Disk Queue Depth Value

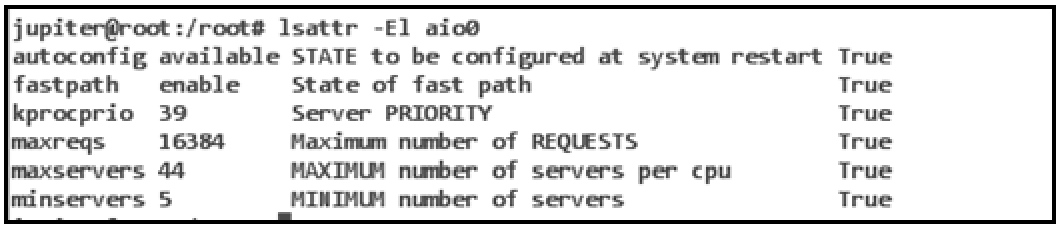
This is the most recommended action when getting Sqfull warnings from disks shown by Iostat. Increasing the disk queue depth might provide some performance improvements. Disk queue depth value determines how many requests the disk drive will queue at any time. This value can be changed from the default value to values from 1 to 256. But this value is specified by Storage disk supplier. Changing value without consulting to vendor will be an unsupported action.

IBM is supporting a depth of 20 for disks of high-end storage system. So there is nothing to do besides testing higher values. But testing these values in production and watching the effects is not possible. And when a higher value will be set, there is another risk that IBM can unsupport these values. Being unsupported by IBM on IBM servers is not appreciatable by anyone.

5.3.3 Asynchronous I/O

Asynchronous I/O (AIO) allows a program to process I/O and continue execution of other works, while other I/O operations are carried out in parallel by the operating system. Because Oracle applications often require multiple servers and user processes at the same time, they take advantage of AIO to overlap program execution with I/O operations. AIO is used with Oracle on the AIX operating system to improve system performance.

When doing I/O to a file system of AIO, each AIO operation is done by an AIO server. Thus, the number of AIO servers limits the number of parallel AIO operations in the system. The first number of servers started at boot time is set by the minservers parameter, which has a default value of one. As more concurrent AIO operations needed, additional AIO servers are started up to limit of maxservers value. The maxservers parameter default value is 10. For Oracle the default values for minservers and maxservers are too small and need to be increased. As seen from Figure 5.4, there values were set to 5 and 44 (minserver and maxserver) for a single processor.



```
jupiter@root:/root# lsattr -El aio0
autoconfig available STATE to be configured at system restart True
fastpath   enable    State of fast path                       True
kprocprio  39        Server PRIORITY                          True
maxreqs    16384     Maximum number of REQUESTS               True
maxservers 44        MAXIMUM number of servers per cpu        True
minservers 5         MINIMUM number of servers                True
```

Figure 5.4 : Number of AIO Server

In addition to tuning minservers and maxservers for AIO, maxreqs will also need to be tuned. The maxreqs value specifies the maximum number of asynchronous I/O requests that are waiting in queue for processing. This value is also set to 16384 before problem.

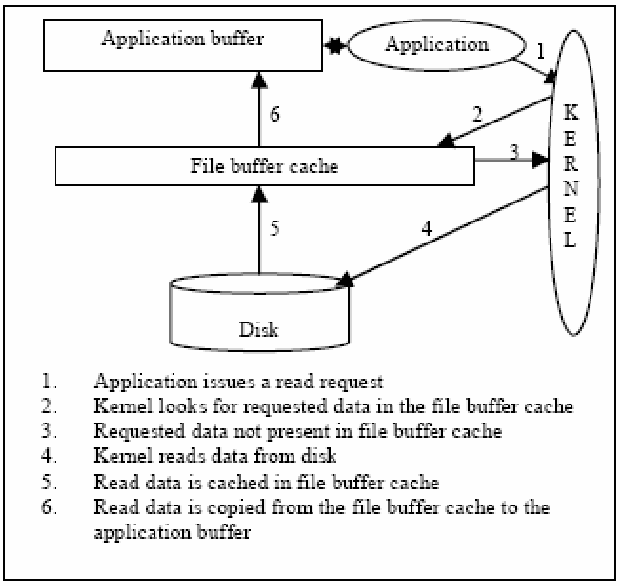
After research's it is decided to set the values of maxservers to 100 AIO per processor. Server has 30 virtual processors so the total value of AIO servers is 3000. A low maxservers value can limit the rate at which AIO requests are completed, thus this setting is resulting in an increase in the I/O requests during a period of heavy I/O activity.

5.3.4 Fs Cache Parameters

Tuning Operating System parameters is the most complex and difficult part. Because most of the parameters depend on other parameters and relation between them can't easily seen before changing the value.

First research is done in the memory and Fs cache tuning parameters, because AIX and Oracle uses file system cache which besides on system memory. When a process wants to access data from a file, the operating system brings this data into main memory, where the process can examine it, change it, and request the data to be saved to disk. The data can also be directly read from disks for each request but the response time and throughput would be poor due to slow disk access times. So the operating system works to minimize the number of disk accesses by storing a copy of data in main memory, which is called the file buffer cache (Kashyap, Olszewski, Hendrickson, 2003).

On a file read request, the file system attempts to read the data from the buffer cache. If the data does not exist in the buffer cache, it will be read from disk and cached in the buffer cache. Similarly, writes to a file are cached so that future reads can be done without a disk access. The use of a file buffer cache can be very effective when the cache hit rate is high. All of this read-write hierarchy can be seen in Figure 5.5.


Application buffer
Application
1
2
3
4
5
6
KERNEL
File buffer cache
Disk
1. Application issues a read request
2. Kernel looks for requested data in the file buffer cache
3. Requested data not present in file buffer cache
4. Kernel reads data from disk
5. Read data is cached in file buffer cache
6. Read data is copied from the file buffer cache to the application buffer

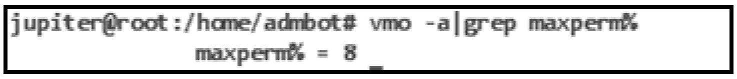
Source: Kashyap, Olszewski, Hendrickson, 2003, Improving DB Performance with Aix CIO

Figure 5.5 : Working Mechanism of File System Buffer

The goal of setting these values are preventing computational memory from paged-out to paging space. Because when data is paged-out, it will have to paged-in from paging space in the future, which would impact system performance poorly. Protecting computational memory is mostly important for applications that maintain their own data cache like Oracle.

5.3.4.1 Maxperm% Parameter

Maxperm% value specifies the upper point where the page stealing algorithms steals only file pages (Lynch, 2005). AIX divides memory either as persistent or working area. Persistent area includes file cache and executables. Working are includes the database. Target amount of memory for persistent storage when the system is paging is the “maxperm” setting. System cannot use more memory for persistent area above maxperm (Darmawan et al, 2003)



```
jupiter@root:/home/admbot# vmo -a|grep maxperm%
                  maxperm% = 8 _
```

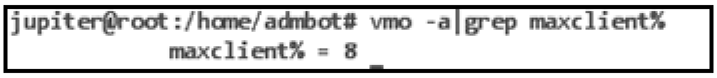
Figure 5.6 : Maxperm% Value

As seen in Figure 5.6, Maxperm% value was set to 8 percent of real memory. After 8 percentage of RAM occupied by persistent area, page-replacement algorithms steals only file pages. So the oracle cache files and other oracle files are being moved to paging space which is much slower than main memory.

By setting Maxperm% value to 80 percent of real memory, persistent memory can use up to 80 percent of real memory for its file buffer cache. But when executables need more memory file pages in memory discarded.

5.3.4.2 Maxclient% Parameter

Maxclient% parameter specifies the maximum percentage of memory that can be used for caching client pages (Lynch, 2005). The “maxclient%” value sets the maximum amount of memory used by file systems. The “maxclient” value is a “hard” limit, which is always enforced.



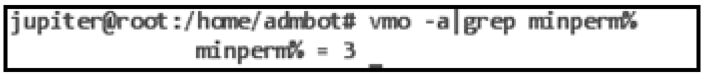
```
jupiter@root:/home/admbot# vmo -a|grep maxclient%
                maxclient% = 8 _
```

Figure 5.7 : Maxclient% Parameter

In our structure it was set to 8 percent of real memory (Figure 5.7), but it will be changed to 80 percent of real memory. So, persistent memory can use up to 80 percent of memory with maxperm% parameter. Within this area, the file system cache is set to use all available area by setting maxclient% to 80 percent. After maxperm% value is reached page stealing algorithms steals file pages.

5.3.4.3 Minperm% Parameter

If percentage of memory used by file pages falls below minperm, page-replacement steals both file and computational pages. This value indicates the critical position. It was set to 3 percent of real memory seen in Figure 5.8.



```
jupiter@root:/home/admbot# vmo -a|grep minperm%
                  minperm% = 3 _
```

Figure 5.8 : Minperm% Parameter

Recommended values are; (Saad, 2006)

i. If physical memory is equal or less than 32G, minperm%=5 percent.
ii. If physical memory is greater than 32G and less than 64G, minperm%=10 percent.
iii. If physical memory is greater than 64G, minperm%=20 percent.

Since DB server has 51 GB of memory, it must be set to 10 percent of real memory.

5.3.4.4 Lru_repage_scan Parameter

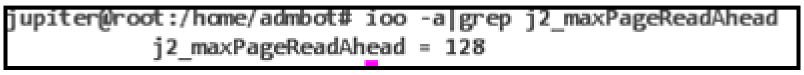
This parameter sets VMM that, what type of memory it should steal. The default setting is 1 this means in a case of memory shortage page stealers begins to steal non file page files in memory. So it needs to be changed in order to steal file pages.

When VMM needs more memory, lrud daemon starts to seek for memory. Then Lrud daemon will make a determination to steal which memory type can be stealable. This determination is made based on some parameters, but the key parameter is lru_file_repage parameter. When lru_file_repage is set to 1 (defaults) the VMM will decide to steal either memory type or just file memory. When the lru_file_repage is set to 0, VMM will only steal file pages in memory. It is also recommended to set to 0, where Oracle is working.

5.3.5 LVM Parameters

5.3.5.1 j2_maxPageReadAhead Parameter

This parameter specifies the maximum number of pages to read ahead when processing a sequentially accessed file on Jfs2 file system (Kashyap, Olszewski, Hendrickson, 2003). It can reduce the time because redo logs are being read and written to Disaster Recovery. This value comes with default value of 128 as seen in Figure 5.9. In most Oracle production systems, random reads are much higher than sequential reads and writes, so it makes no sense to change this value.

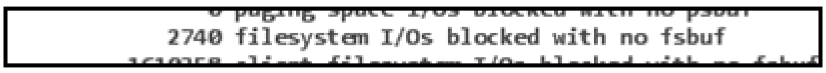


```
jupiter@root:/home/admbot# ioo -a|grep j2_maxPageReadAhead
               j2_maxPageReadAhead = 128
```

Figure 5.9 : j2_maxPageReadAhead Parameter

5.3.5.2 Numfsbufs Parameter

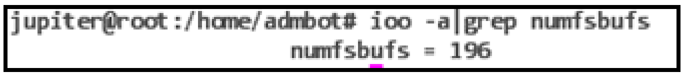
Fsbufs are pinned memory buffers, used to hold I/O requests in the file system layer (ibmsystemsmag, 2006). When a read or write request comes to Logical Volume Manager (LVM) and the fsbufs queue is full, the VMM must wait for a free fsbufs so it puts the request to the VMM wait list. If an fsbufs has become available the request is waken. As seen from Figure 5.10, 2740 I/O requests are ignored or putted to queue because of inability of fsbuffers.



```
         0 paging space I/Os blocked with no psbuf
      2740 filesystem I/Os blocked with no fsbuf
   1610358 client filesystem I/Os blocked with no fsbuf
```

Figure 5.10 : Fsbufs Bottleneck

If there are many random or large I/O requests to a file system, it might become a bottleneck at the file system level while waiting for free fsbuffers. So it is important to change this value according to needs. It had already set to 196 like in Figure 5.11.



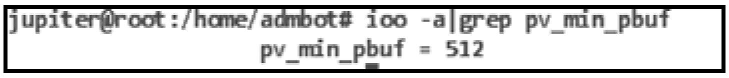
```
jupiter@root:/home/admbot# ioo -a|grep numfsbufs
                      numfsbufs = 196
```

Figure 5.11 : Numfsbufs Parameter

This value can be set between 128 to 2048, by testing higher values. It is decided to test a value of 512. If again blocked I/O errors starts, a higher value will be tested.

5.3.5.3 Pv_min_pbuf Parameter

Pbufs are pinned memory buffers used to hold I/O requests at the logical volume manager layer (ibmsystemsmag, 2006). Pv_min_pbuf parameter sets the number of pbufs to add when a disk is added to a volume group. So when a new FS needed to be created on new disks, existing pbufs will be insufficient. New pbufs must be created.



```
jupiter@root:/home/admbot# ioo -a|grep pv_min_pbuf
                    pv_min_pbuf = 512
```

Figure 5.12 : Pv_min_pbufs Parameter

This value was set to 512 by default as in Figure 5.12. But when io_blocked errors become to appear, the number of pbufs is not enough with default value of 512. So changing this value to 1024 is better to test.

5.4 DISCUSSION OF THE RESULTS

One of the most useful benefits of Iris monitoring is its capability to do trend analyses. Performance data will be stored for 5 years period. So after some major changes, improvements in performance can be proven with graphs.

Now let’s see the performance benefits of tunings. 03/04/2008, 04/04/2008 is the days before tuning and 24/04/2008, 25/04/2008 are the days after tuning. These are the same days of weeks. (Thursday and Friday) There is no special difference between these two days (Special means new banking promotions or new applications). From the below figures (Figure 5.13, Figure 5.15, Figure 5.17, Figure 5.14, Figure 5.16, Figure 5.18), it can be seen that; daily workload is nearly same for the selected days.

For Figure 5.13, Ticket number means the total number of banking operations that completes in success. The data in Figure 5.13 represents values between 03/04/2008 (Thursday) and 04/04/2008 (Friday). There is an average of 171 banking operations per minute.

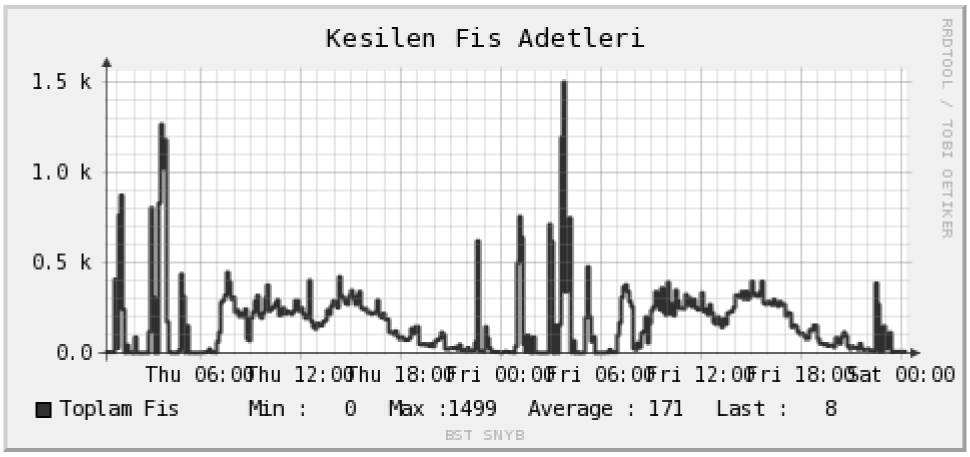


Figure 5.13 : Number Of Tickets Between 03/04/08 – 04/04/08

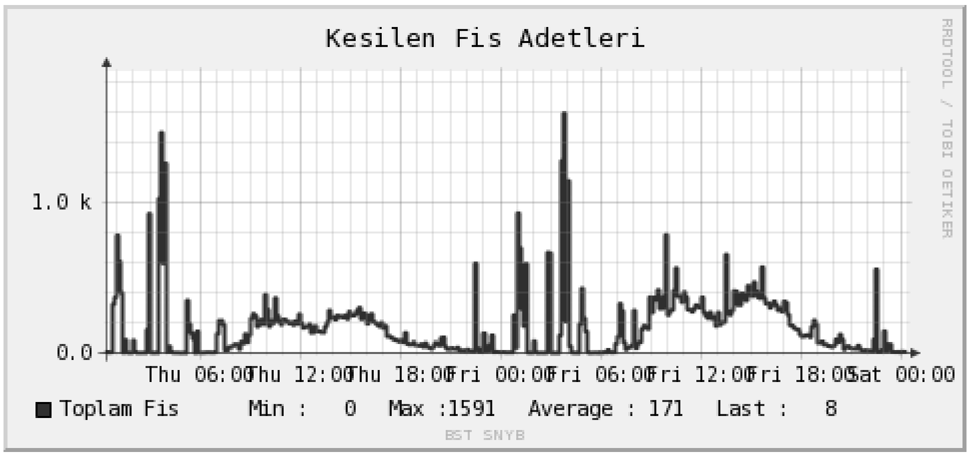


Figure 5.14 : Number Of Tickets Between 24/04/08 – 25/04/08

In Figure 5.15 there is an average of 171 operations per minute between 24/04/2008 (Thursday) and 25/04/2008 (Friday). Number of tickets in target two days is the same. (Same number of banking operations on selected dates)

Session numbers is the number of clients connected to banking operations with new web applications. Number of NGBS clients is the sum of users with old applications. E-bank clients are connected with E-Bank application clients. In Figure 5.15, the average of sessions per minutes are; 1630, 67, 120 between 03/04/2008 and 04/04/2008.

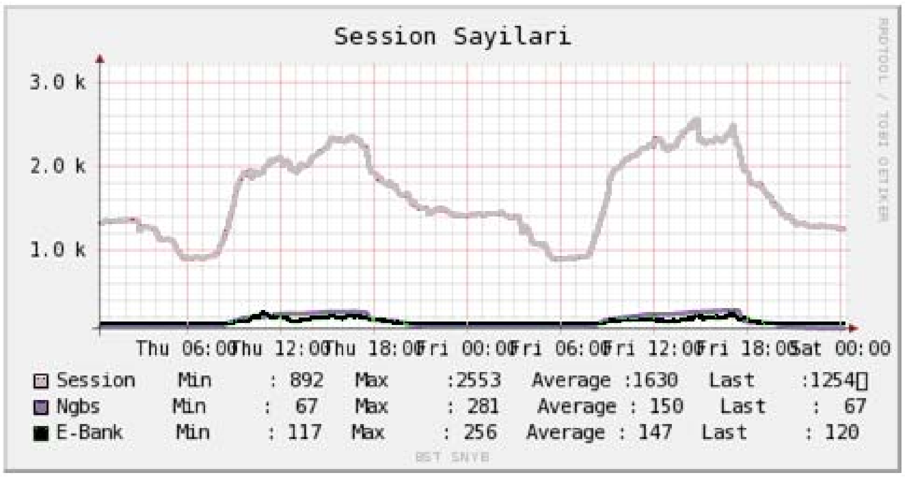


Figure 5.15 : Number Of Sessions Between 03/04/08 – 04/04/08

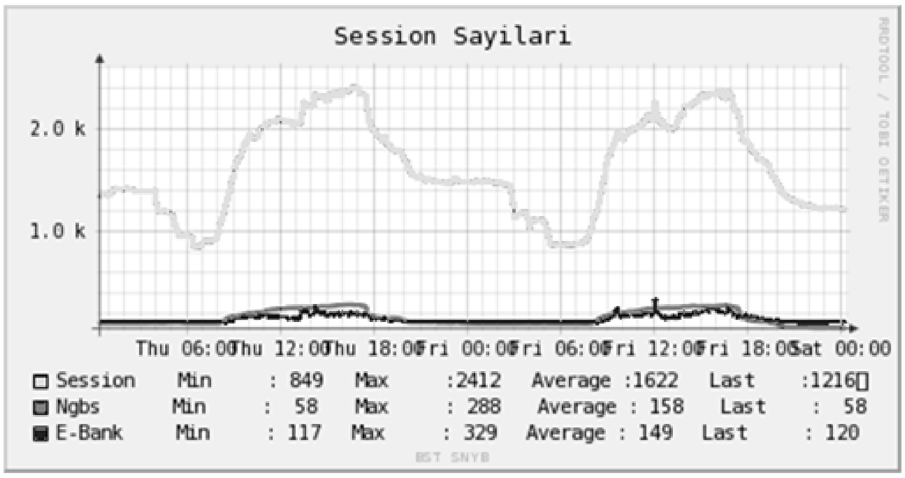
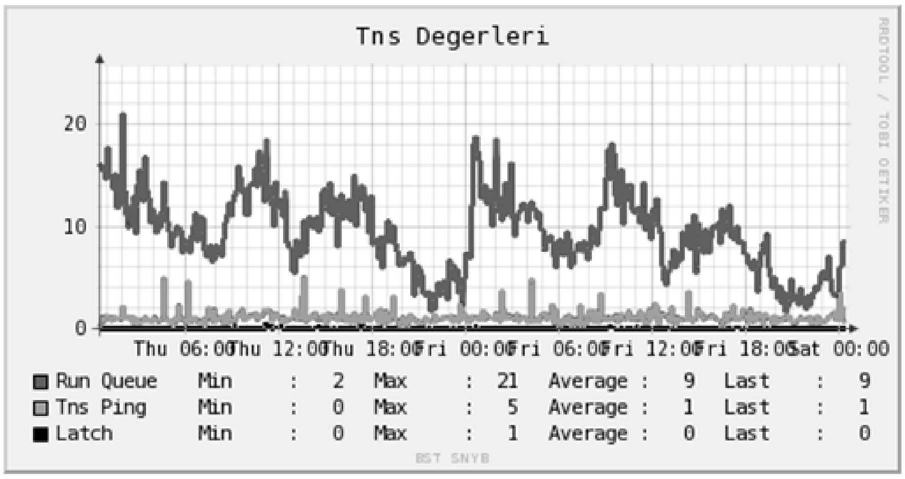


Figure 5.16 : Number Of Sessions Between 24/04/08 – 25/04/08

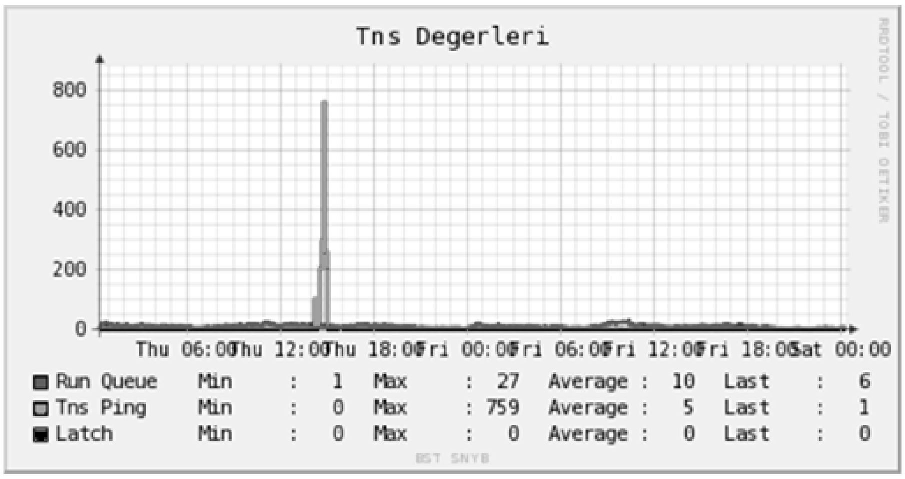
In Figure 5.16, there is an average of 1622, 158, 149 sessions per minute between 24/04/2008 and 25/04/2008.

In Figure 5.17 and Figure 5.18 some Oracle related parameters will be considered. In Figure 5.17 there is an average of 9 run queues per minute.



```
■ Run Queue   Min   :  2   Max   :  21   Average :   9   Last   :  9
▨ Tns Ping    Min   :  0   Max   :   5   Average :   1   Last   :  1
■ Latch       Min   :  0   Max   :   1   Average :   0   Last   :  0
```

Figure 5.17 : Oracle Performance Data Between 03/04/08 – 04/04/08



```
■ Run Queue   Min   :  1   Max   :  27   Average :  10   Last   :  6
▨ Tns Ping    Min   :  0   Max   : 759   Average :   5   Last   :  1
■ Latch       Min   :  0   Max   :   0   Average :   0   Last   :  0
```

Figure 5.18 : Oracle Performance Data between 24/04/08 – 25/04/08

In Figure 5.18 there is a run queue of 10. (Because of the peak value in “Tns Ping” value, graph is not balanced)

As it can be seen in above figures selected two work days are nearly same for banking operations. Now with the help of Iris monitoring system performance improvements on selected days will be shown.

5.4.1 CPU Usage Improvements

In this part, improvements in CPU usage will be displayed in details. For dates 03/04/2008 (Figure 5.19) and 04/04/2008 (Figure 5.21), these are the graphs before tuning. For the Figure 5.19 and Figure 5.20, a bit improvement in CPU usage can be seen.

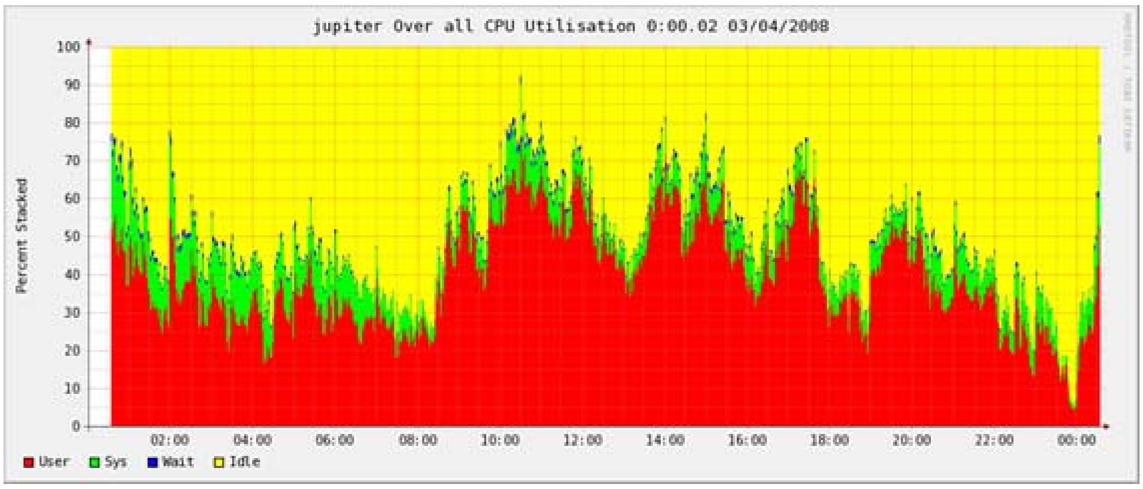


Figure 5.19 : Cpu Usage Graph for 03/04/2008

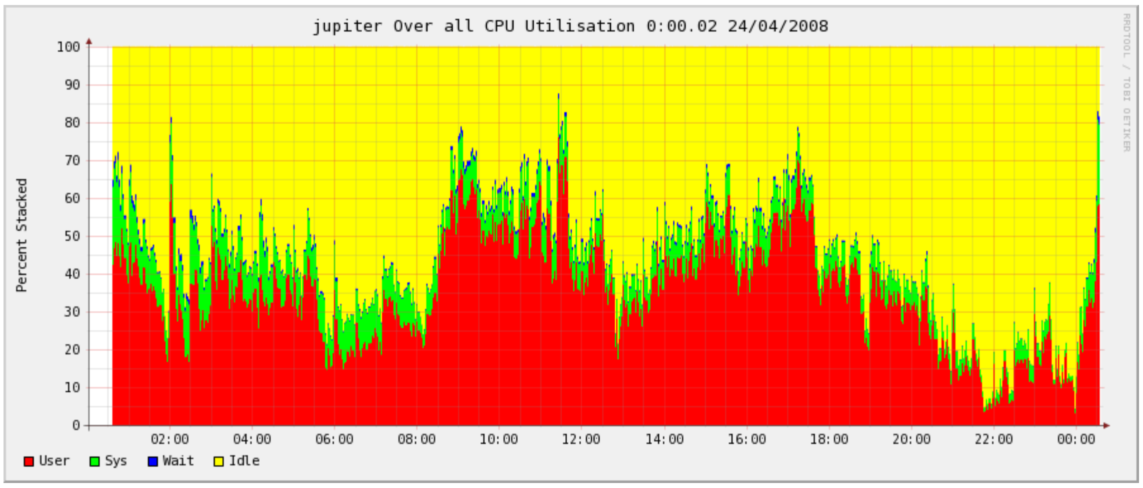


Figure 5.20 : CPU Usage Graph For 24/04/2008

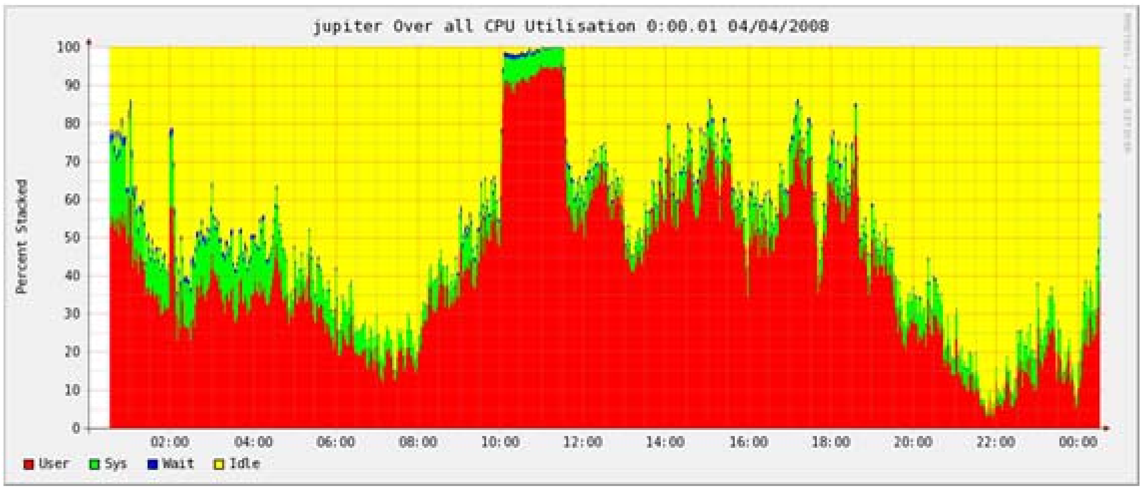


Figure 5.21 : CPU Usage Graph For 04/04/2008

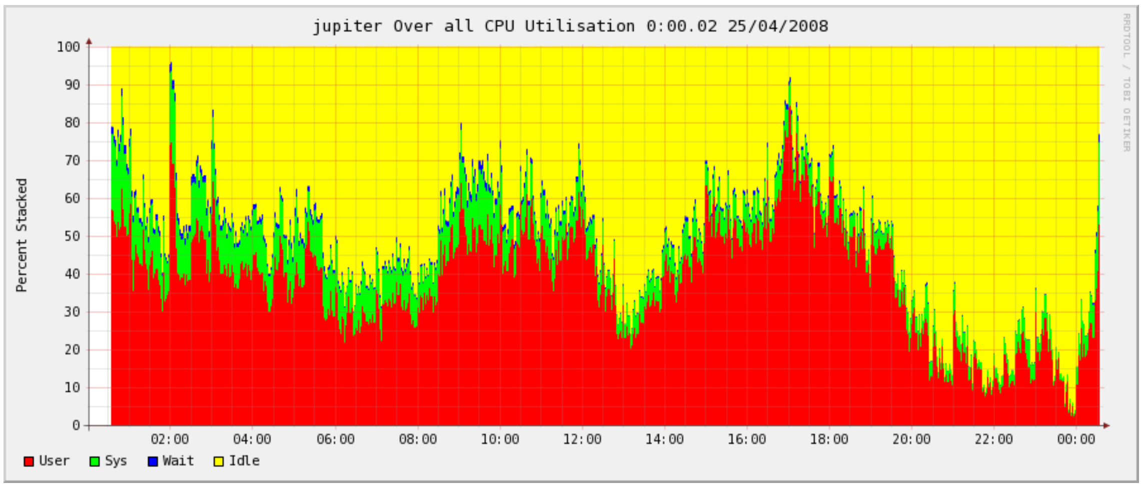


Figure 5.22 : CPU Usage Graph For 25/04/2008

CPU usage improvements can be clearer in Figure 5.21 and Figure 5.22. In 04/04/2008 system has a peak value of 100 percent, which means that CPU can't cope with processes. But in 25/04/2008 after tuning made to system, working characteristics is more constant and there are no high peak values.

Beside these improvements, high improvements on CPU usage should not be expected. Because, after tunings made, it is expected that disk queues won't be full again, so they can process more requests which costs more CPU usage. More CPU usage can be expected, but because of the disk queues were handled more efficiently, no more peaks are expected.

5.4.2 Memory Usage

Most of major changes after tuning have happened in memory usage. Before tuning, although system has 51 GB memory, nearly 25 percent or this memory wasn't being used. They remain unused for all time. After tunings system is going to use this area for file system caching.

All of the files used by system are cached by operating system for future needs. So when a read request comes, system first looks in this memory region. Because memory is much faster than disks system will gain huge performance improvement.

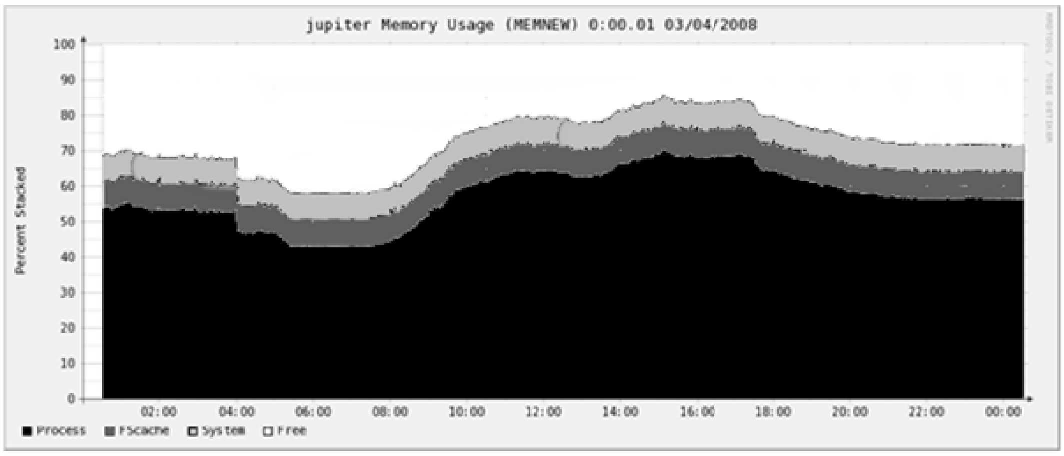


Figure 5.23 : Memory Usage Graph for 03/04/2008

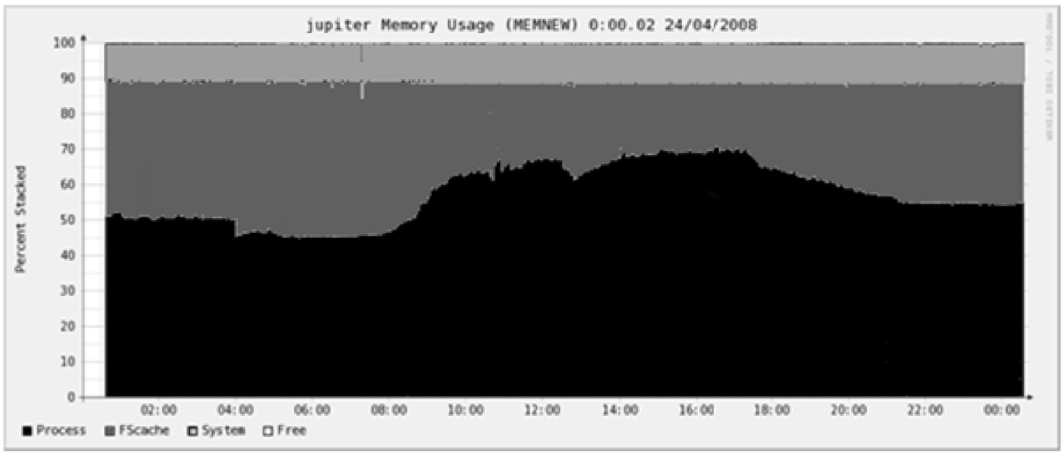


Figure 5.24 : Memory Usage Graph for 24/04/2008

In Figure 5.23 before tunings, as told before nearly 25 percent of memory were not used. In peak times 85 percent of memory was used for both persistent and working sets of memory.

But after tunings in Figure 5.24, unused memory regions in Figure 5.23 were used for persistent memory. FS caches which is limited to 10 percent is now limited to 80 percent, so if processes frees memory, they started to use for FS cache. Although no free memory is displayed in system, in a case of memory need, these FS caches will be discarded.

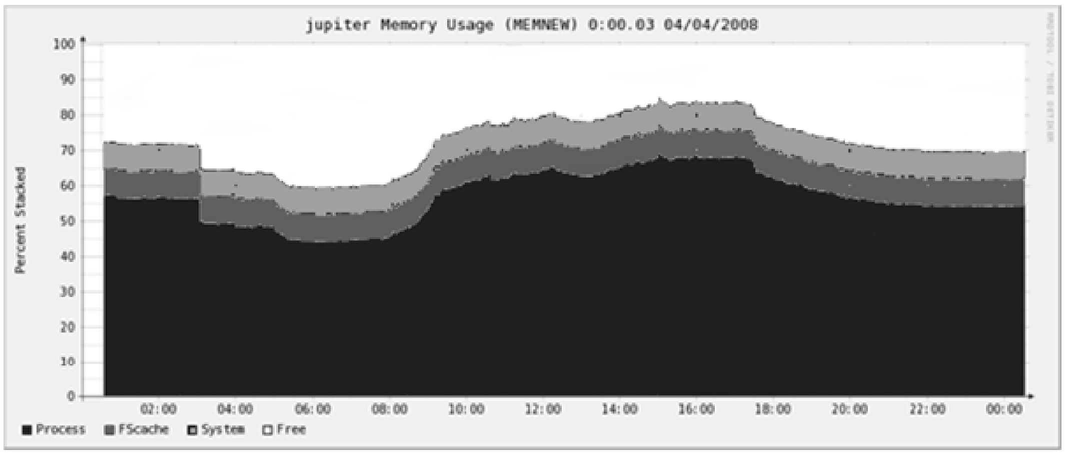


Figure 5.25 : Memory Usage Graph for 04/04/08

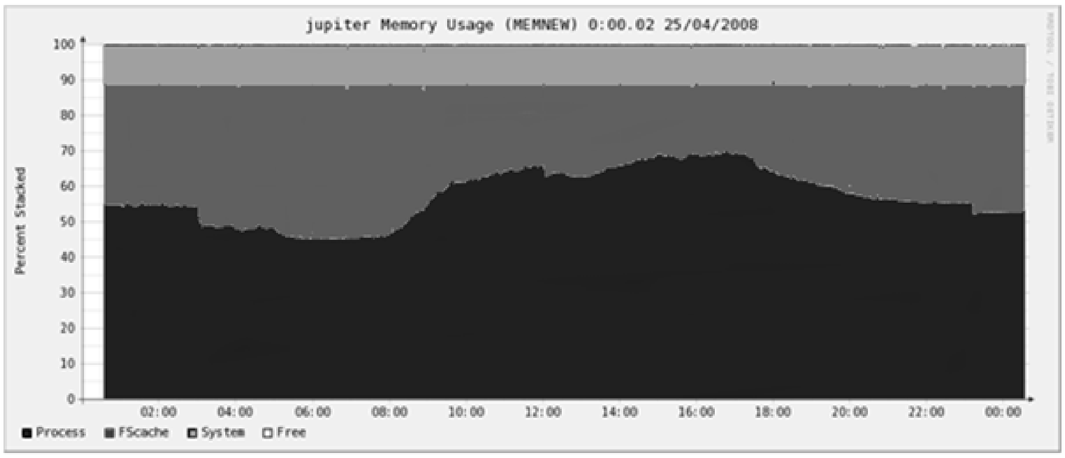


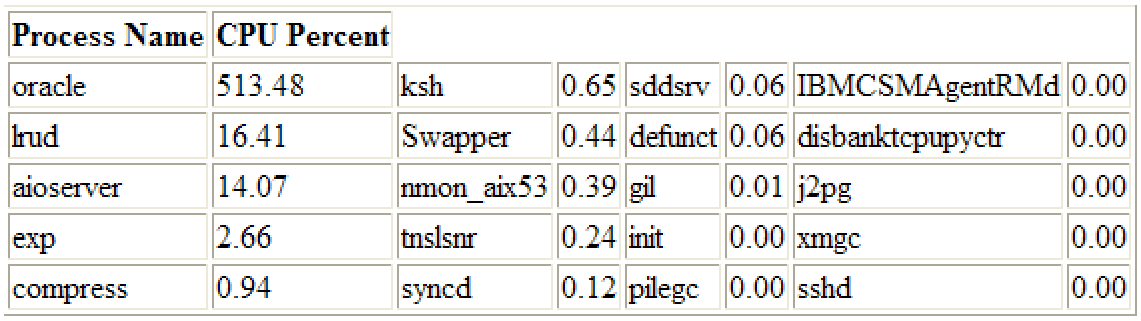
Figure 5.26 : Memory Usage Graph for 25/04/08

Again in Figure 5.25 system is not tuned for FS cache usage. But in Figure 5.26 FS cache usage it tuned for best performance.

5.4.3 Top Processes

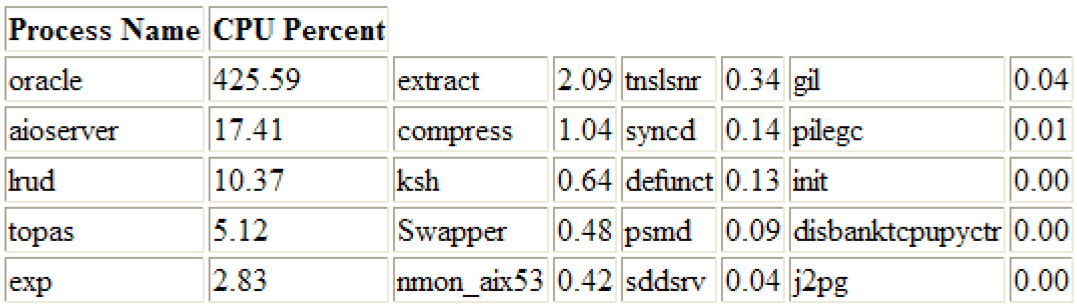
By the help of file system caching Oracle processes won't get service queue full errors from disk subsystem, or they won't wait in disk queue. So they don't wait on processor which results in better respond times and lower CPU usages. Below charts are the top processes for selected days. (These data's are gathered from Aix Nmon tool)

In Figure 5.27 Oracle processes gets 513.48 percent per minute, where 100 percent is equals to 1 CPU (So it uses 5.13 CPU per minute). The second top process is lrud process. It is a background daemon of VMM. It is responsible for scanning in memory pages and freeing up memory in real memory. Because there is a lot of free memory it's getting 16.41 percent per minute (0.16 CPU per minute).



Process Name	CPU Percent						
oracle	513.48	ksh	0.65	sddsrv	0.06	IBMCSMAgentRMd	0.00
lrud	16.41	Swapper	0.44	defunct	0.06	disbanktcpupyctr	0.00
aioserver	14.07	nmon_aix53	0.39	gil	0.01	j2pg	0.00
exp	2.66	tnslsnr	0.24	init	0.00	xmgc	0.00
compress	0.94	syncd	0.12	pilegc	0.00	sshd	0.00

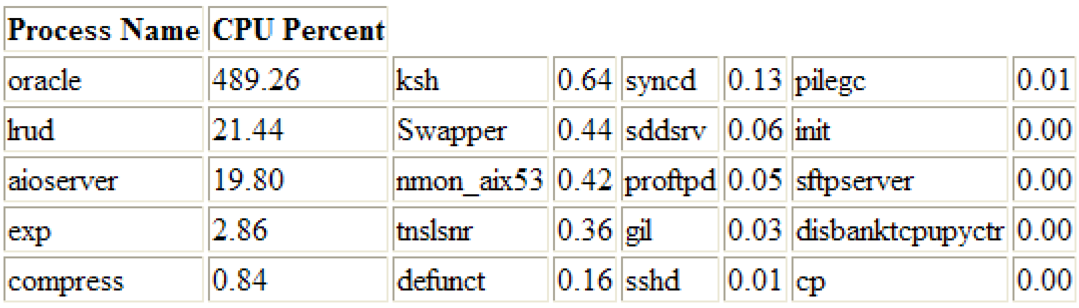
Figure 5.27 : Top Processes for 03/04/08



Process Name	CPU Percent						
oracle	425.59	extract	2.09	tnslsnr	0.34	gil	0.04
aioserver	17.41	compress	1.04	syncd	0.14	pilegc	0.01
lrud	10.37	ksh	0.64	defunct	0.13	init	0.00
topas	5.12	Swapper	0.48	psmd	0.09	disbanktcpupyctr	0.00
exp	2.83	nmon_aix53	0.42	sddsrv	0.04	j2pg	0.00

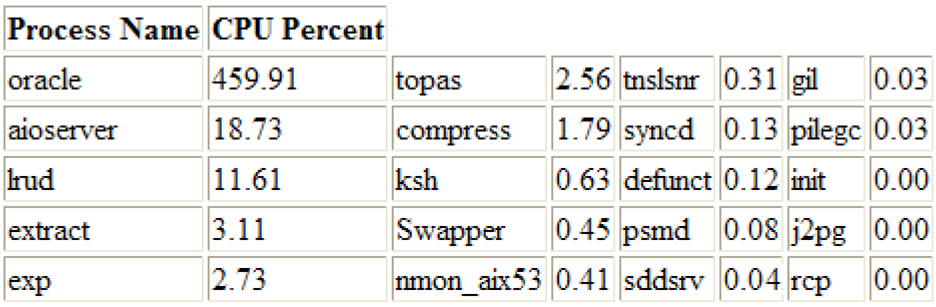
Figure 5.28 : Top Processes for 24/04/08

As discussed in Chapter 5.4.1 (CPU Improvements) CPU usage is expected to decrease. In Figure 5.28 Oracle process's CPU usage is decreases to 425 percent which is equal to 4.25 CPU per minute. The difference from Figure 5.27 is 0.88 CPU per minute which equals to 7 percent of all processing power. The second top process is process of AIO servers. It also expected because there is no more bottleneck in disk subsystem. So they are working more efficient.



Process Name	CPU Percent						
oracle	489.26	ksh	0.64	syncd	0.13	pilegc	0.01
lrud	21.44	Swapper	0.44	sddsrv	0.06	init	0.00
aioserver	19.80	nmon_aix53	0.42	proftpd	0.05	sftpserver	0.00
exp	2.86	tnslsnr	0.36	gil	0.03	disbanktcpupyctr	0.00
compress	0.84	defunct	0.16	sshd	0.01	cp	0.00

Figure 5.29 : Top Processes for 04/04/08



Process Name	CPU Percent						
oracle	459.91	topas	2.56	tnslsnr	0.31	gil	0.03
aioserver	18.73	compress	1.79	syncd	0.13	pilegc	0.03
lrud	11.61	ksh	0.63	defunct	0.12	init	0.00
extract	3.11	Swapper	0.45	psmd	0.08	j2pg	0.00
exp	2.73	nmon_aix53	0.41	sddsrv	0.04	rcp	0.00

Figure 5.30 : Top Processes for 25/04/08

Also in Figure 5.29 and Figure 5.30 Oracle process's CPU usage is decreased. lrud process's Cpu usage is decreased because minperm% is increased to 10 percent of real memory.

5.4.4 Paging Space

Although there are 25 percent free memory, operating system paged out memory pages from memory to satisfy free memory percentage, because memory parameters were not optimum tuned. As seen in Figure 5.31, although it seems to be very little paging in working hours, paging spaces are placed in internal SCSI disks which are much slower than SAN disks or physical memory. By changing the minperm and maxperm, it tells operating system that paging is forbidden since there is a huge need to real memory. And when a need occurs, first computational Oracle pages will be discarded. Because they can be discarded no paging is needed.

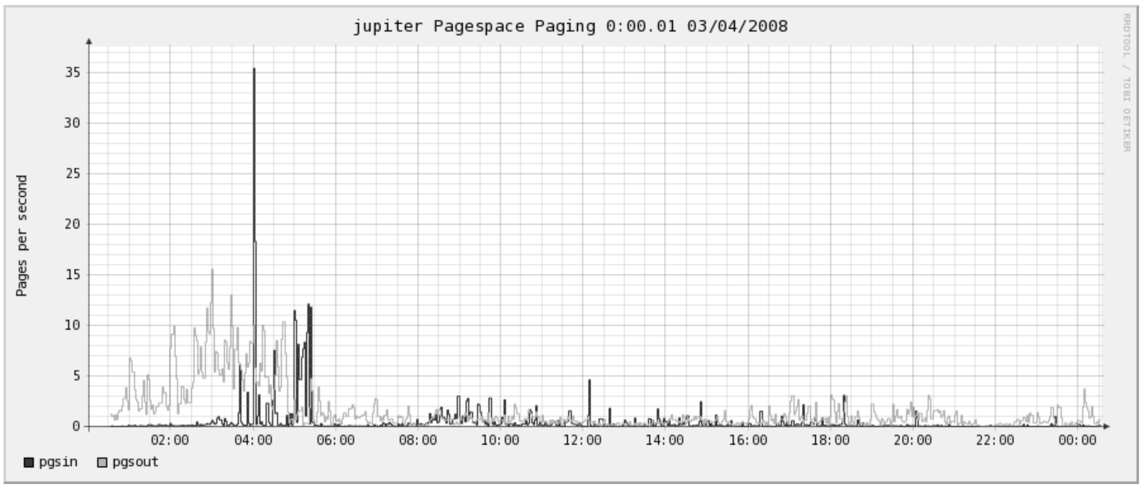


Figure 5.31 : Paging Space Usage for 03/04/08

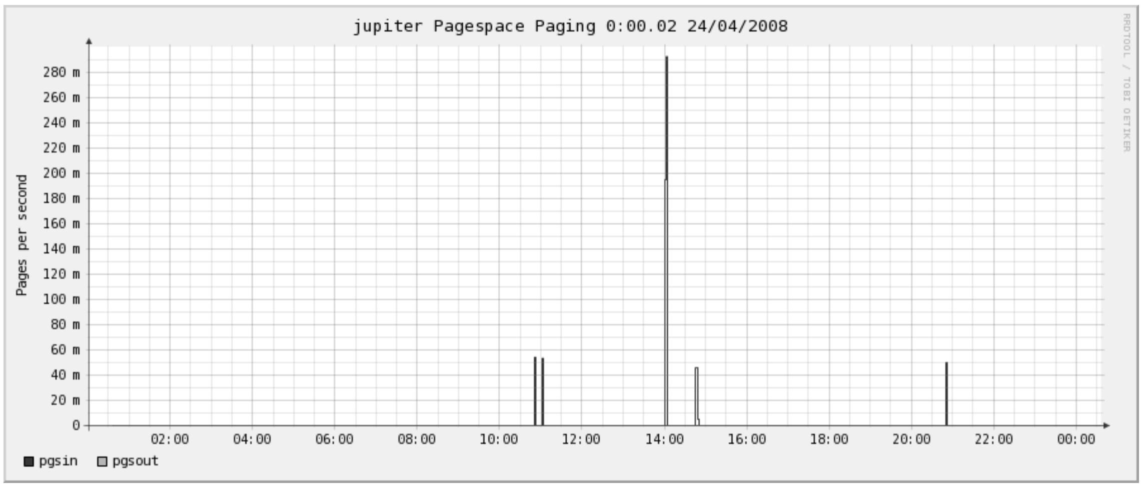


Figure 5.32 : Paging Space Usage for 24/04/08

After tunings made to system as seen from Figure 5.32, there is no need to make paging. no paging is done.

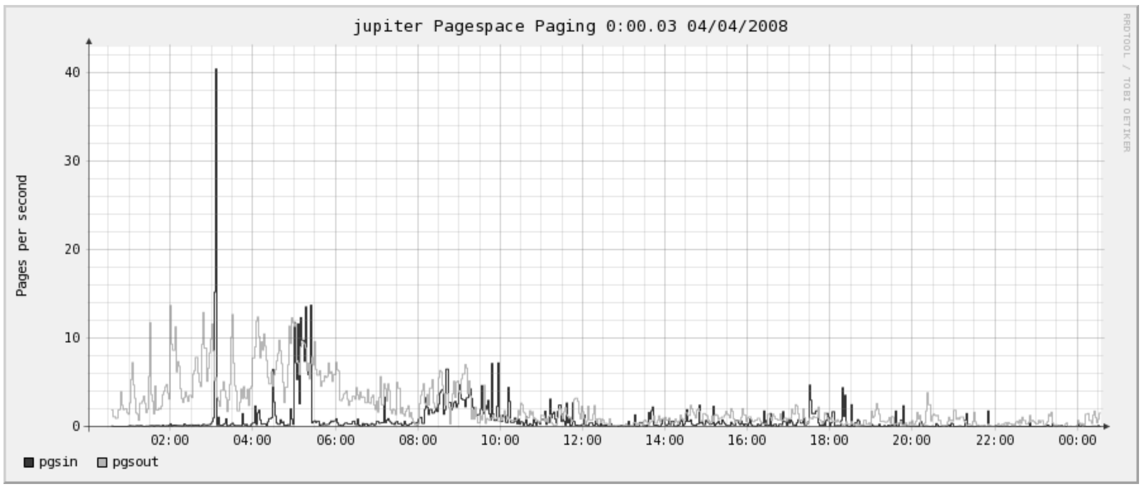


Figure 5.33 : Paging Space Usage for 04/04/08

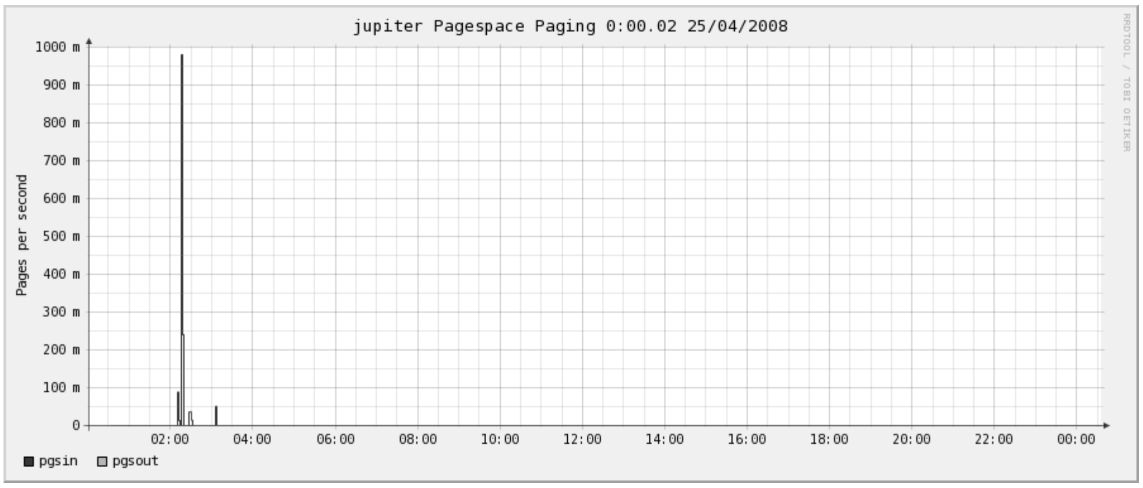


Figure 5.34 : Paging Space Usage for 25/04/08

Again in Figure 5.33 and Figure 5.34, it can be seen paging activity before and after tunings made. It is also important to not to use paging to save CPU run queue.

5.4.5 Improvements in Oracle DB

Although system performance improvements seen in Iris graphs, performance improvements can be tracked from Oracle performance reports.

Table 5.1 Oracle Performance Improvements

Before Tuning		**After Tuning**		**Improvement**
RW		**RW**		**RW**
AVG_IOWAIT_TIME	70,02	AVG_IOWAIT_TIME	39,148	%-44,0
IOWAIT_TIME	1686,04	IOWAIT_TIME	944,517	%-43,9
CIO		**CIO**		**CIO**
AVG_IOWAIT_TIME	32,537	AVG_IOWAIT_TIME	30,381	%-6,62
IOWAIT_TIME	786,489	IOWAIT_TIME	734,597	%-6,59

In Table 5.1;

AVG_IOWAIT_TIME is the number of hundredths of a second that a processor has been waiting for I/O to complete, averaged over all processors (Oracle.com, 2009). So when FS are mounted with RW option the average wait time is 70 milliseconds. But after tunings made the average wait time drops to ~39 milliseconds which shows ~ 44 percent improvements in responses.

IOWAIT_TIME is the number of hundredths of a second that a processor has been waiting for I/O to complete, totaled over all processors (Oracle.com, 2009). There are again ~ 43 percent improvements in waiting times for all operations.

CIO mount is another option but as considered in previous chapters, it is not available to use in current configuration. It is always a rule that, CIO mount option gives the available best performance with Oracle. In planned tests, AVG_IOWAIT_TIME for processes gets 32 milliseconds before tuning and ~30 milliseconds after tunings made.

As a contrast, the AVG_IOWAIT_TIME after tunings made to system in RW mount option, the processes gets nearly same values as mounted with CIO option. This is another proof that, improvements makes a better performanced system as shown with Iris.

REFERENCES

Books

Bueche, E., Harris, C., 1999. *Documentum Performance and Tuning*. US: Documentum

Darmawan, B., Kamers, C., Pienaar, H., Shiu, J., 2003. *AIX 5L Performance Tools Handbook*. US: IBM Redbooks

Hayashi, K., Ji, K., Lascu, O., Pienaar, H., Schreitmueller, S., Tarquino, T., Thompson, J., 2005. *AIX 5L Practical Performance Tools and Tuning Guide*. US: IBM Redbooks

Kashyap, S., Olszewski, B., Hendrikson, R., 2003. *Improving Database Performance with AIX Concurrent I/O*. US: IBM

Saad, B., J., 2006. *VMM Tuning Tip: Protecting Computational Memory*. US: IBM

Hoogenboom, P., J., 1991. *System Performance Advisor: An Expert System For UNIX System Performance Management*. US: The University of Utah

Periodical Publications

Lynch, J., 2005. *Tuning AIX Commands for JFS2. IBM Systems Magazine*. April-May, 2005, ss.1-2

Lynch, J., 2006. *Tuning a Perfect Note. IBM Systems Magazine*. August-September, 2006, ss.1-5

Speier, G., 2005. *An Application to Provide UNIX Performance Analyses, Bottleneck Determination.*

Finkel, R., A., 1997. *Pulsar: An Extensible Tool for Monitoring Large Unix Sites.*

Miller, T,. Stirlen, C., Nemeth, E., 1993. *Satool: A system Administrator's Cockpit, an Implementation*, In Proceedings of Seventh Systems Administration Conference, ss.119 - 129.

Knop, W., M., Dinda, A., P., Schopf, M., J., 2001. *Windows Performance Monitoring and Data Reduction using WatchTower.*

Apisdorf, J., Claffy, K., Thompson, K., 1996. *Flexible, Affordable, High Performance Statics Collection.*

Spring, N., Hayes, J., 1991. *A Distributed Resource Performance Forecasting Service For Metacomputing.*

Other Publications

Apache. http://en.wikipedia.org/wiki/Apache
[cited January 2009]

Apache. http://www.apache.org
[cited January 2009]

Ganglia. http://en.wikipedia.org/wiki/Ganglia
[cited November 2008]

Ganglia. http://ganglia.info/
[cited November 2008]

IBM Information Center. http://publib.boulder.ibm.com/infocenter/
[cited January 2009]

Response Time. http://en.wikipedia.org/wiki/Response_time_(technology)
[cited November 2008]

TK. http://en.wikipedia.org/wiki/Tk_(framework)
[cited December 2008]

UNIX Shell. http://en.wikipedia.org/wiki/UNIX_shell
[cited January 2009]

Oracle Data Guard. http://en.wikipedia.org/wiki/Oracle_Data_Guard
[cited January 2009]

Vbscript. http://en.wikipedia.org/wiki/Vbscript
[cited January 2009]

Vbscript. http://www.SearchEnterpriseDesktop.com
[cited January 2009]

Windows Performance. http://www.homenetworkhelp.info/index.php?pg=podcast-2007
-07-30

[cited January 2009]

AVG_IOWAIT_TIME. http://download-uk.oracle.com/docs/cd/B19306_01/ server.102
/b14237/dynviews_2010.htm
[cited January 2009]

CURRICULUM VITAE

Name Surname	: Hakan HALİSÇELİK
Address	: 49.Ada Mimoza2 Sitesi 5.Blok No:5 Ataşehir / Kadıköy / Istanbul / Türkiye
Birth Place / Year	: Ankara / 1981
Languages	: Turkish (native) - English
High School	: Yüce Science High School -1999
BSc	: Başkent University - 2004
Name of Institute	: Institute of Science
Name of Program	: Computer Engineering
Work Experience	: September 2004 – Present Assistant Manager - Servers and Storage Management Systems Fortis Bank