Basic Monitoring of I/O on AIX

This document can be found on the web, www.ibm.com/support/techdocs

Version Date: 12 03, 2008

Katharina Probst
Dirk Michel

IBM Boeblingen Lab, Germany
SAP on Power Systems Development @ SAP
© Copyright IBM Corp. 2008
Table of contents

Basic Monitoring of I/O on AIX ................................................................. 1
Version Date: 12 03, 2008........................................................................ 1
Abstract......................................................................................................... 3
Conventions..................................................................................................... 4
Introduction .................................................................................................... 5
  Data Collection for I/O Monitoring............................................................... 6
  Tools overview.......................................................................................... 7
Analysis of AIX® clients ............................................................................. 13
  Is there an I/O issue?................................................................................ 13
  Memory: VMM, Paging and Page Replacement ........................................ 14
  CPU......................................................................................................... 19
  Local storage: File system, Logical and Physical Disks............................. 27
  Network ................................................................................................. 31
  NFS Client ............................................................................................. 35
  Virtual I/O Client.................................................................................... 40
Analysis of AIX® Server ............................................................................. 41
  Network ................................................................................................. 42
  Memory and CPU ................................................................................... 42
  Logical and Physical Discs ...................................................................... 42
  NFS Server ............................................................................................ 43
  Virtual I/O Server ................................................................................ 43
Summary........................................................................................................ 44
Resources...................................................................................................... 45
About the authors........................................................................................ 45
Trademarks and special notices ................................................................. 46
Abstract

This document is intended to provide a detailed and example based description on basic I/O monitoring for people who are new to performance analysis on AIX®. The focus is to give background information on I/O flow through AIX® systems, a list of best practice approaches, rules of thumb and examples for I/O performance analysis in a step by step guide.

By restricting the scope to the basic tools shipped with AIX 5.3L on POWER5 machines this document does not cover all monitoring areas such as storage. The reason is that the tools used for that purpose do not fit into the approach covered by this document.

For people familiar with I/O monitoring this document can be used as a reference of basic tools or to look for alternative approaches on how to monitor I/O on AIX®. Advanced performance analysts might consider, based on their experience, not to keep to the proposed step by step guide and using advanced tools which require a higher level of understanding traces, or collecting and interpreting data to get a more detailed picture. Having the ability to work with the advanced tools will allow them to also monitor the areas not covered here.
Conventions

Although performance is a relative measure this document provides a few numbers helping estimate if a value is high or low. The values always depend on the workload or the hardware and must not be seen in any different context as stated. These rules are marked as follows:

Rule of thumb (’for what’):
Text

Using PERFPMR the data collection is slightly different compared to gathering the information manually via the command line or with other tools. Therefore hint-boxes on how to use PERFPMR and how to find the needed information look like the following:

Hint (PERFPMR):
Text

Beside general rules and PERFPMR hints there are other tips and tricks in this document which are similar to the ones above:

Hint (’for what’):
Text
Introduction

Performance is not an absolute characteristic, it always depends on the type of workload, on the used resources, for example storage type, and the customer’s expectation. Therefore this paper uses in many cases relative figures like high/low since in most cases no absolute values can be given. To assign absolute values for a specific system or system landscape it has to be monitored regularly to detect increasing values for that specific setup of hardware and workload.

I/O in general means input/output in the sense that everywhere in a computer system, between computer systems or even between a computer system and users an input is followed by an output. I/O is not a single event. It always is part of a flow of different I/O operations.

For example a user accessing wikipedia to search for information, where the initial input is “Search for: I/O” and the final output is the article about I/O, produces different types of I/O: The web server gets the input to ask the data base (DB) server to search for the content. As an output the web server sends an SQL request over the fabric to the DB. The input to the DB server is a SQL statement. When searching in the database the server generates I/O on the CPU, memory, and so on. The final output of the DB is the content of the article which is sent back over the fabric to the web server and then via the web to the user. Again all these final steps are generating I/O.

Figure 1 Example I/O environment

The scope of this paper is restricted to I/O on AIX® systems, between AIX® systems and between AIX® systems and other parts in the landscape. Figure 1 shows on a high level view the most
common resources causing I/O. The components covered in this document are an AIX® client and server, an NFS client and server, and a VIO client and server each connected by a network.

Beside the mentioned parts I/O operations occur as well over a fabric, special adapters or on storage systems which is beyond the scope of this paper. The recommendation for these areas is to collect PERFPMR data regularly to be able to provide a good performance history of the system for advanced analysis by specialists.

System configurations consist of chains of client-server dependencies. That means a client can become a server and vice versa. For example the Virtual I/O Server (VIOS) in Figure 1 is the server for the client above but has as well the client credentials for the storage. This should make it obvious that the following Figure 2 has to be seen as an abstract top-down model for every client and bottom-up model for every server in the chain of client-server dependencies. In order to use this model this document is divided into two parts: Analysis of AIX® clients and Analysis of AIX® servers.

![Diagram showing client: top-down and server: bottom-up](image)

*Figure 2 Client top – down, Server bottom-up approach*

**Data Collection for I/O Monitoring**

This part gives a proposal on how to collect data. We recommend to start gathering data on the client beginning with the application like a data base or an SAP® application showing the performance issue. Then we follow on the client side downwards to the operating system and
finally through the adapters or interfaces to the server side. Since data collection itself may cause noticeable load on a system exhibiting bad performance already, it is highly recommended to collect one set of data, do the analysis and then do the next step.

Recommended chain of data collection and analysis:

1. Application
   - SAP: for example transactions like OS07n can be used.
   - Data base: the DB statistics provide first hints. In case a trace is found, it should be tracked down to the file if possible before further investigations are made.
   - ...

2. List of any recent changes of:
   - Hardware
   - Network and SAN configuration
   - OS
   - Tuning
   - Applications
   - ...

3. Operating System:
   For optimal monitoring information data should be collected before, during, and after the issue shows up on the server, as well as on client side at the same time. It is not recommended to always request the whole data, since data collection can be a lot of work and in some cases the additional load could bring the system finally down. Data collection can be done with provided script based tools or by calling AIX® tools manually.
   - PERFSAP as documented in the SAP note 1170252 is a well defined tool to collect data on SAP systems running on AIX®.
   - PERFPMR is a well defined collection of different AIX® performance collecting tools which is provided and supported by IBM®.
   - Collecting kernel traces in case PERFPMR can not be applied.
   - Command line execution of selected tools.

4. Information of the network’s setup:
   - Adapters
   - Switch configuration and speed
   - Used cables
   - How storage is attached

**Tools overview**

This chapter provides a collection of AIX 5.3 tools most relevant for I/O performance monitoring used in this document. For each tool the main usage as well as a little example and annotations if
appropriate are provided. The tools often can be used for further purposes and get regular enhancements due to new features. To get information beyond this document a good reference is: http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.doc/infocenter/base/aix53.htm

The tools are divided into basic and advanced monitoring tools with the focus on the basic ones delivered with AIX 5.3. The reasons to mark tools as advanced are: the output are traces which require experience to interpret them or the output can be easily misunderstood.

vmstat
Reports virtual memory statistics and is not the first choice for I/O related CPU information. The CPU utilization reported by vmstat is valid for shared partitions; vmstat reports usr, sys and idle relative to physical processor consumed (pc) and entitlement consumed (ec). When the pc is less than ec vmstat will show usr, sys, and idle. For uncapped partitions when pc is greater ec vmstat will report only usr and sys.

iostat
Is used to get the enhanced CPU statistics not delivered by vmstat. It reports CPU statistics, asynchronous I/O (AIO) and I/O statistics for the entire system, adapters, tty devices, disks and CD-ROMs. It is a lightweight CLI (Command Line Interface) to filemon, without the possibility to get detailed information about logical volumes and seek times (although some useful information is available using the –f option showing a filesystem utilization report).

filemon
Monitors the performance of the file system, and reports the I/O activity on behalf of logical files, virtual memory segments, logical volumes, and physical volumes. Since filemon is a very heavy tool it can not be run in every case and only for a very short time.

tuncheck
Validates a specified tunable file (tuncheck [ -r | -p ] -f Filename ). All tunables listed in the specified file are checked for range and dependencies. If a problem is detected, a warning is issued. This tool is valuable when the problem is tracked down to a file and after every change of a tunable file.

nfso
Can be used to configure and view NFS attributes in NFS client-server analysis situations.

netpmn
Is used to find hot files or processes by looking for unusual response times. However it has more capabilities such as:

- CPU usage
- Network device driver I/O
- Internet socket calls
- NFS I/O
- Calculated response times and sizes associated with:
- Transmit and receive operations on the device driver level.
- All types of Internet socket read and write system calls.
- NFS read and write system calls as well as NFS remote procedure call requests.

**vmo**

Can be used to configure or display current or next boot VMM (Virtual Memory Manager) tuning parameters. Whether the command sets or displays a parameter is determined by the accompanying flag. The `-o` flag performs both actions. It can either display the value of a parameter or set a new value for a parameter. In this paper it is used as the basic AIX® tool for memory tuning on AIX®.

**lsps**

Displays characteristics of a paging space (or all paging spaces). This includes:

- Paging-space name
- Physical-volume name
- Volume-group name
- Size
- Percentage of the paging space used
- Whether the paging-space is active, inactive or automatic

For NFS paging spaces, the physical-volume name and volume-group name is replaced by the host name of the NFS server and the path name of the file used for paging.

**ftp**

Can be used to perform a memory to memory copy between two LPARs. Therefore it is a great tool to analyze issues on the network connectivity since this excludes side effects of not network related I/O due to slow disks, CPU, etc. The used command is:

```bash
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
```

**entstat**

Displays the statistics gathered by the specified Ethernet device driver. The user can optionally add the device-specific statistics to the device generic statistics. If no flags are specified, only the device generic statistics are displayed.

**netstat**

Traditionally, netstat is more a problem determination instead for performance measurement tool. However, the netstat command can be used to determine the amount of traffic on the network to ascertain whether performance problems are due to network congestion.

The netstat command displays information regarding traffic on the configured network interfaces, such as the following:

- The address of any protocol control blocks associated with the sockets and the state of all sockets.
The number of packets received, transmitted, and dropped in the communications subsystem.

Cumulative statistics per interface.

Routes and their status.

topas

Gives hints if any resources are short. Topas reports selected statistics regarding activities on the local system and as well a cross-partition view. Also a recording functionality is provided including the preprocessing tool topasout to generate different views. Following is a list of the monitored resources in topas:

- Processor
- Memory
- Network interfaces
- Physical Disks
- Workload Manager Classes
- Processes
- Cross partition view recording

PERFPMR

Is a script calling a number of AIX® monitoring tools to collect a set of the most common performance information provided by IBM® and published on the IBM® homepage. A basic concept of PERFPMR is to collect two data sets `command`.before and `command`.after. This is used for tools providing snapshot data where the difference over time is essential. Beside formatted output it collects also traces which can be preprocessed as shown in the following example:

Start the script ...

```
PERFPMR.sh -x trace.sh 5
```

... and post-process all data at once:

```
PERFPMR.sh -r
```

As an alternative the following can be done to get the same output:

Steps of PERFPMR.sh -x trace.sh 5 to collect the trace:

```
bin/trace -p -r PURR -k 492,10e,254,116,117 -f -n -C all -d -L 20000000 -T 20000000 -ao trace.raw
sleep 5
trcstop
```

After the trace is collected single parts can be preprocessed as following:

```
trcnnm > trace.nn
gensyms > trace.syms
genstrings > gennames.out
```

To create a trace report out of a given trace:

```
trcstop
```
errpt (advanced)
Generates a report of logged system errors. At first glance errpt seems to be a very basic tool. But in some cases wrong conclusions can be drawn and therefore it is marked as advanced.

Beside checking the errpt output directly on the system's commandline it can be generated out of a trace collected for example from PERFPMR. The following preprocessing step explains how to generate the output out of a trace file:

```
errpt -y errtmplt -i errlog -a > errpt_a.out
```

iptrace (advanced)
By default, iptrace provides a detailed, packet-by-packet description of the LAN activity. The option -a allows exclusion of address resolution protocol (ARP) packets. Other options can narrow the scope of tracing to a particular source host (-s), destination host (-d), or protocol (-p). Due to the fact that the iptrace daemon can consume significant amounts of processor time, usage requires to be as specific as possible when describing the packets to be traced.

ipreport (advanced)
Generates a trace report from the specified trace file created by the iptrace command. To obtain a detailed, packet-by-packet description of the LAN activity, the iptrace daemon (see above) and the ipreport command is required.

ipfilter (advanced)
Extracts specific information from an ipreport output file and displays the information in a table format. The operation headers currently recognized are: udp, nfs, tcp, ipx, icmp, atm. The ipfilter command has three different types of reports:

A single file (ipfilter.all) that displays a list of all selected operations. The table displays packet number, time, source and destination, length, sequence number, ack number, source port, destination port, network interface, and operation type.

Individual files for each selected header (ipfilter.udp, ipfilter.nfs, ipfilter.tcp, ipfilter.ipx, ipfilter.icmp, ipfilter.atm). The overall information is the same as ipfilter.all.

A file nfs.rpt that reports on nfs requests and replies. The table contains: transaction ID number, type of request, status of request, call packet number, time of call, size of call, reply packet number, time of reply, size of reply, and elapsed milliseconds between call and reply.

svmon (advanced)
Provides data for an in-depth analysis of memory usage. It is more informative, but also more intrusive, than the vmstat and ps commands. The svmon command captures a snapshot of the current state of memory. For evaluation purposes it is essential to have snapshots over time to get a timeline how memory is used.
trace (advanced)

Helps to isolate system problems by monitoring selected system events or selected processes. Events that can be monitored include: entry and exit to selected subroutines, kernel routines, kernel extension routines, and interrupt handlers. trace can also be restricted to tracing a set of running processes or threads, or it can be used to initiate and trace a program.

trcrpt (advanced)

Used to format a report from a given trace log. The following example shows how trcrpt can be used on base of the log trace.raw:

```
trcrpt -C all -r trace.raw > trace.tr
trcrpt -C all -t trace.fmt -n trace.nm -O timestamp=1,exec=on,tid=on,cpuid=on trace.raw > trace.int
trcrpt -C all -t trace.fmt -n trace.nm -O timestamp=1,exec=on,tid=on,cpuid=on,PURR=on trace.raw > trace.int
trcrpt -C all -r trace.raw.lock > trace.tr.lock
```

tprof (advanced)

Reports processor usage for individual programs or the system as a whole. This command is a useful tool to analyze a Java, C, C++, or FORTRAN program that might be processor-bound to determine the most processor consuming sections of the program.

The tprof command can charge processor time to object files, processes, threads, subroutines (user mode, kernel mode and shared library) and even to source lines of programs or individual instructions. Charging processor time to subroutines is called profiling and charging processor time to source program lines is called micro-profiling.

An example based on trace data looks like following:

```
tprof -skje[R] -r trace
```

splat (advanced)

The Simple Performance Lock Analysis Tool post-processes AIX® trace files to produce kernel lock usage reports. It also produces pthread mutex read-write locks, and condition variables usage reports.

An example based on a trace looks like following:

```
splat -i trace.tr.lock -n trace.syms -d a -o splat.out
```

curt (advanced)

Takes an AIX® trace file as input and produces statistics related to processor (CPU) utilization and process/thread/pthread activity. It works with both, uniprocessor and multiprocessor AIX® traces if the processor clocks are properly synchronized.

Two examples based on a trace:
```
curt -i trace.tr -n trace.syms -t -p -e -s -r PURR -o curt.out
```
**Analysis of AIX® clients**

This chapter introduces the usage of the basic AIX® tools to analyze AIX® clients. Figure 3 shows the top-down client analysis used in this document for every client without taking the service provided by a server like NFS into account. At the end of this chapter, special-purpose clients attached to a NFS or VIOS server are handled separately.

Is there an I/O issue?

As stated already in the introduction every I/O issue in the client server chain has a direct effect on the client. So it is sufficient to check the client for I/O shortage. For that initial check AIX® provides iostat which will be compared to vmstat to clarify the difference.

The CPU statistics of *vmstat* shows only I/O waits for the assigned CPU leading to the situation that a fully utilized CPU can never show I/O waits since it never has to wait for input. Also the I/O wait parameter is in percent of CPU time, which is hard to interpret when CPUs are shared. That means on the one hand that the system can run into I/O issues while *vmstat* does not show any waits. On the other hand it can show high I/O wait, but the system, depending on the workload is entirely fine. For example a backup can cause high I/O wait time since it uses a lot of disk I/O, but hardly processing time.

---

*Figure 3 General client analysis steps*
The advantage of `iostat` in comparison to `vmstat` is that it shows also I/O problems when the system uses all CPU and is therefore of higher quality. The `iostat` tool reports CPU statistics, asynchronous I/O (AIO) and I/O statistics for the entire system, adapters, tty devices, disks and CD-ROMs.

**Rule of thumb (iostat):**

To determine if there is an issue the `tm_act` parameter in `iostat` has to be checked. It heavily depends on the workload when `tm_act` is to be interpreted as an I/O issue. A backup running with full load can set the active time up to acceptable 100%, whereas other workloads such as on data base server will crash way earlier.

**Hint (PERFPMR):**

PERFPMR data contains the same information by calculating the delta values between `vmstat_s.p.before` and `vmstat_s.p.after`. Some delta values can also be found in `monitor.sum`.

### Memory: VMM, Paging and Page Replacement

The Virtual Memory Manager (VMM) in AIX® distinguishes two types of pages. For all pages containing open files to write updates into, called file pages (FP), the VMM provides the page replacement mechanism. For all pages containing information from running programs, called computational pages (CP), the VMM paging mechanism is applied. The core of the AIX® VMM tuning is to define:

- How much memory can be used in total.
- Two thresholds: the first one defines when to write back only FP and the second one when the system starts also the paging of CP.

For the system it is important to keep the executables running. Therefore the VMM tries to keep CP in memory or uses paging space to access them fast. The FP are always written back to the storage. If FP would be paged out to the paging space, the used files would be blocked for all other applications till they are written back to the file system. This would have a deep performance impact if the FP are needed by concurrent applications. Therefore CP can be paged out and FP are written back in case memory has to be freed (Figure 4).

![Figure 4 Paging versus page replacement](image-url)

---

Basic Monitoring of I/O on AIX

14
The tool to start with is `vmstat` since it shows the page-in (pi) and page-out (po) as well as the replaced files (fr). In addition the `vmo` command can be used to check the system for correct VMM tuning parameters.

**Determine if the system is paging computational pages**

When determining if a system might be short on memory or if some memory tuning needs to be done, the `vmstat` command can be used for monitoring the system over a reasonable period of time while the performance issue occurs. The columns `pi` and `po` indicate the number of paging space page-ins and page-outs per second. If the values are constantly non-zero, there might be a memory bottleneck whereas having occasional non-zero values is a principle of virtual memory and by that no issue.

**po (page-out):**

When a system writes CP to the paging space, it is not necessarily an issue since the processes do not stop as long the amount of free memory (`fre`) is greater than the vmo tuning parameter `minfree` (`minfree` will be introduced later). But it is an indication for an upcoming issue when this page has to be paged-in later.

**pi (page-in):**

Every time a CP has to be paged-in means that a running application is blocked until the CP is written back into memory from the paging space. This is the final performance issue since paging-out alone does not influence performance, only the process of getting the data back is critical.

**Rule of thumb:**

For general workload a paging space usage above 5% is often an issue. This rule has to be taken carefully since the amount of memory, the size and the utilization of the paging space depends on the application. This rule can for example not be applied for SAP systems since there a usage of 30% of the paging space can be fine if the sizing for paging space and real memory is correct.

The following is a portion of `vmstat` of a paging system:

In the given example the `minfree` value of 960 per memory pool is reached because page replacement (grey box) occurs. If the free list (`fre`) reaches zero running programs will be blocked and can’t run until page replacement frees FP to provide space for page-ins of CP.

```
#vmstat 1

System configuration: lcpu=4 mem=1024MB ent=0.20

kthr   memory          page          faults         cpu
------- ----------- -------------------- ------------------------
        r   b  avm  fre  re  pi  po  fr  sr  cy  in  sy  cs  us  sy  id  wa  pc
        ec
   1  0  144059  61061  0  0  0  0  0  0  14  154  149  1  3  97  0  0.01
   1  0  144057  61063  0  0  0  0  0  0  12  285  145  1  3  96  0  0.01
   2  0  219258  7451  0  0  0  21589 202426  3  20  10251 212  74  26  0  0.49
   243.6
```

Basic Monitoring of I/O on AIX
Another tool is /lsp -a. It shows the percentage of used paging space in order to determine if it is big enough.

Hint (PERFPMR):
The lsp information can be retrieved by calculating the delta value of lsp.before and lsp.after

**Determine if the system replaces file pages**

The vmstat displays page replacement in the fr (freed) column showing how many pages have been freed by the LRU daemon. Page replacement can be determined by looking on the following values in vmstat:

- The LRU deamon starts always to replace file pages as soon as
  
  \[ \text{fre} \text{ in } \text{vmstat} < \text{mminfree} \text{ in } \text{vmo} - \text{a} \]

- The ratio between sr (scanned pages) and fr (freed pages) defines how many file pages the LRU had to scan in order to find one that could be freed. It also depends on how many file pages in comparison to computational pages are currently in memory. If a high rate is constantly reached, tuning can force to page the CP sooner to improve performance. Also the later described tining parameter lru_file_repage has an effect on the behavior.

**Rule of thumb (ratio sr/fr):**

*Pre AIX 6.1:*

The ratio sr/fr gives an indications if the I/O performance is fine. With AIX® versions before 6.1 the ratio depends on the used vmo tuning.

- vmo tuning allows 90% file pages: \( \text{sr/fr} < 1.2 \)
- vmo tuning allows 50% file pages: \( \text{sr/fr} < 2.1 \)
- vmo tuning allows 10% file pages: \( \text{sr/fr} < 9.1 \)

*AIX 6.1:*

With LRU enhancements in AIX 6.1 the ratio of 1 is reached as long as the system is fine. This is due to a free list maintained by the VMM.

**Hint (PERFPMR):**
The PERFPMR dataset provides the vmstat statistics. Again there are two snapshots where to calculate the delta out of.
Tuning Parameters

vmo

The `vmo` command can be used to change and check the VMM settings. When experiencing paging or page replacement the first thing to check are the VMM settings, which can be displayed with the `vmo -a` command. The objectives in VMM tuning are to ensure the following:

- Ensure that any activity having critical response-time objectives can always get the page frames it needs from the free list.
- Ensure that the system does not experience unnecessarily high levels of I/O caused by permanent stealing of pages to expand the free list.

This section covers the six most important VMM tuning parameters: `minfree`, `maxfree`, `numperm`, `maxperm`, `minperm` and `lru_file_repage`. Further tuning of vmo settings heavily depends on deep system analysis and by that it is considered as advanced tuning.

**Hint (PERFPMR):**

The system settings displayed by the `vmo -a` command are listed in `config.sum` and in `mempools.out` (per memory pool).

**Hint (SAP):**

The recommended SAP tuning parameters for AIX® can be found in OSS note 1048686. IBM’s general recommendation is to use the default settings coming with AIX 6.1.

**minfree and maxfree**

The Virtual Memory Manager (VMM) maintains a list to keep track of free real-memory page frames (free list). These page frames are available to hold virtual-memory pages needed to satisfy a page fault. This allows page frames to be supplied to requestors immediately, without forcing them to wait for page steals and the accompanying I/O to complete. The boundaries used for that purpose are called `minfree` and `maxfree`. When the number of pages on the free list falls below `minfree`, the VMM begins to steal pages and adds them to the free list. The VMM continues to steal pages until the free list contains the number of pages specified by the `maxfree` parameter. Real memory is split into evenly sized memory pools based on the number of CPUs and the amount of RAM. Each memory pool has its own `minfree` and `maxfree` values. Prior to AIX 5.3 the `minfree` and `maxfree` values shown by the `vmo` command are the sum of all memory pools. Starting with AIX 5.3 and later, the values shown by `vmo` are per memory pool. The number of memory pools can be displayed with `vmo -L mempools`.

The default values for `minfree` and `maxfree` on AIX 5.3 and later are sufficient for most common workloads. However, some workloads such as SAP with heavy cached file system activity require increasing the values for `minfree` and `maxfree` to prevent the situation where the free list drops to 0 free pages.

**numperm, maxperm, minperm and lru_file_repage**

These tunables define whether to steal CP, FP or both. If the number of permanent FP in memory is less than the number specified by the `minperm%` parameter, the VMM steals frames from either CP or FP, regardless of re-page rates. If the number of permanent FP is greater than the number specified by the `maxperm%` parameter, the VMM steals frames only from FP. Between the two,
the VMM normally steals only FP, but if the re-page rate for file pages is higher than the repage rate for CP, CP are stolen as well.

AIX® has the following method of paging/page replacement regarding the values of numperm, maxperm and minperm (displayed by vmstat -v):

- $numperm < minperm$: LRU steals CP and FP
- $maxperm > numperm > minperm$: LRU steals those with less re-pages, FP preferred
- $numperm > maxperm$: LRU steals only FP

If the lr_file_repage parameter is set to 0, only file pages are stolen if the number of file pages in memory is greater than the value of the minperm parameter.

**Paging due to memory over-commitment or memory leak**

The usage of virtual memory allows more virtual memory to be addressed in total than physical memory is available. Memory over-commitment occurs when applications request from the OS more virtual memory than physically is available. In this case it becomes difficult for the VMM to choose pages for paging because they will probably be referenced in the near future by currently running processes. This can lead to continuous paging in and out, called thrashing. When a system is thrashing, the system spends most of its time with paging instead of executing useful instructions, and none of the active processes make any significant progress.

**Hint (memory overcommitment):**

The `svmon -G` tool can show if the amount of real memory is smaller than the amount of virtual memory.

A memory leak for example happens when an application do not release memory correctly. An indicator for memory leaks gives the `svmon -G` tool. It shows in case of a memory leak a constant growth of memory usage. Tuning in this case can not help and the application has to be fixed to release memory correctly.

Other things that can lead to paging or replacement:

- Wrong sizing of physical memory.
- Data base tunable can allow the DB to use up all memory so that not enough memory for the kernel and other applications is left.
- User assigned memory to another LPAR without remembering.
- The queue to the paging space can be a bottle neck.

**Hint (queue size for paging):**

If the number of page outs is higher than operations a disk can physically write per second the system has an I/O issue. Even if po in the `vmstat` is zero the queue can be still full of pending page-outs. This can lead to the situation that page-ins are not possible till CP are paged-out. That means that the process has to wait till the page is finally written to the paging space and back.

**Summary:**

Check if memory is over committed (add more memory if needed):

The `avm` in `vmstat` is bigger than the amount of real memory pages.

More virtual than real memory pages exist. For example `svmon` can be used to check this.
If memory is not over committed:

Tune VMM page replacement to reduce paging.

Check what pages are paged-out to the paging space. For instance FP with the deferred update pages flag will be written to paging space. That means these FP will be blocked for all writes.

Open an AIX® PMR.

**CPU**

CPU influences I/O performance as soon as a constant usage of 100% is reached. This reduces I/O although the I/O flow could be faster if the CPU would be able to handle all incoming requests in time. AIX® has four different ways how to assign CPUs to a partition what has to be taken into account when looking at the performance values.

**Terms and technologies in AIX® and Power Systems**

There are four different ways to assign CPUs to an LPAR which can be used on the same box at the same time. Dynamic LPAR operations allow CPU settings to be changed during runtime. Also energy saving features will add more complexity when turning off or changing the frequency of physical CPUs while the box is running. WPARs will introduce a further virtualization layer of CPU. Therefore this chapter gives a short introduction into the available technology before giving insight into monitoring.

Dedicated LPAR:

Assigning dedicated CPUs to an LPAR is the simplest way and was introduced with POWER4. Dedicated means this LPAR owns whole assigned physical processor core(s) no matter if the LPAR uses the cycles or not. This makes monitoring easy because it is well defined on what amount of CPU the utilization is based on.
Shared LPAR (SPLPAR) capped/uncapped:

The main difference between shared and dedicated is that SPLARs residing on a commonly shared pool of physical processors and have parameters to define how they compete for these processors. Parameters to look at are the entitlement, the amount of virtual CPUs (VCPU), the mode and the weight. The entitlement defines the guaranteed amount of processing time in fractional of whole CPUs. This entitled capacity is shared with other SPLPARs on the same pool as long they are not needed. A capped partition is not allowed to exceed its entitlement, while an uncapped partition is allowed to exceed the entitlement within defined boundaries. These boundaries are the weight, the amount of VCPUs and obliviously by the physical processors. Uncapped SPLPARs with a high weight have advantages in comparison to SPLPARs with a low weight when competing for free resources. Those uncapped partitions are only limited in their ability to consume cycles by the amount of online VCPU. Each VCPU can represent one physical processor at maximum and hence introduces an implicit capping. This type of partition was introduced with the POWER5 architecture.

Hint (SPLPAR monitoring):

When monitoring SPLPARs the interpretation of the utilization depends on the consumed entitlement. If the SPLPAR does not use the whole entitlement an utilization of 100% is normal since the partition can get at any time its entitlement. When looking to those SPLPARs running in an uncapped mode they can get CPU cycles beyond their entitlement if needed. In this case stealing cycles from the shared pool is entirely fine as long this does not impact other partitions. Hence in depth monitoring of uncapped SPLPARs requires often a system wide information gathering.

Dedicated shared LPAR:

A new feature for POWER6, Shared Dedicated Capacity, allows partitions running with dedicated processors to "donate" unused processor cycles to the shared-processor pool. When enabled in a partition, the size of the shared processor pool is increased by the number of physical processors normally dedicated to that partition. This increases the simultaneous processing capacity of the associated SPLPARs. Due to licensing concerns, however, the number of processors an individual SPLPAR can acquire will never be more than the initial processor pool size. This feature provides a further opportunity to increase the workload capacity of uncapped micro-partitions.

Physical, virtual and logical CPU, max CPU and simultaneous multi-threading (SMT)

The physical hardware holds the physical processor cores which can be assigned as a dedicated processor to a Dedicated LPAR and/or into the shared pool. On the partition another layer of CPU virtualization has been introduced called virtual processors, which contain the power of the currently assigned physical CPU and limit the amount of CPU power of uncapped SPLPARs. Finally the VCPUs of a SPLPAR can be split up into two logical CPUs by enabling the SMT feature.
The following figure 5 displays the difference between physical, virtual and logical processors.

**Figure 5 Example of CPU virtualization with PowerVM**

**Monitoring and Tuning**

The basic CPU monitoring tools shipped with AIX® are `vmstat`, `lparstat` and `topas`. The tools are used to display static information about the LPAR and the entire box as well as current values. Due to the flexible setup AIX® provides for allocating processor resources, the static information have to be collected first. Based on the static information the interpretation of the current values of the CPU is possible.

In general CPU tuning is assigning CPUs (physical, virtual or logical) and deciding whether to use SMT or not. All other tuning parameters should be the default as shipped with AIX®.

**vmstat**

The information if the LPAR is running shared or dedicated can be indirectly found with `vmstat`. When running in shared mode the columns `pc` and `ec` appear which are not displayed with dedicated LPARs. Please also look at `vmstat` in the “Tools overview”.

```
# vmstat -w 1 4
System configuration: lcpu=4 mem=4096MB ent=0.40

kthr  memory       page  faults  cpu
    ----  -----------  ---------  ---------  ----
    r b   avm  fre  re  pi  po  fr  sr  cy  in  sy  cs  us  sy  id  wa  pc
```

Basic Monitoring of I/O on AIX
The usage of `vmstat` is already described in “Is there an I/O issue?” chapter.

Hint (vmstat):

The `-w` flag of `vmstat` provides beginning with AIX 5.3 a better layout.

`lparstat`

To look deeper into CPU issues `lparstat` provides views of the static information and current statistics depending on the used flag.

The view `lparstat -i` displays the configuration of the LPAR:

```
#lparstat -i
Node Name : is3015
Partition Name : is3015
Partition Number : 7
Type : Shared-SMT
Mode : Uncapped
Entitled Capacity : 0.40
Partition Group-ID : 32775
Shared Pool ID : 0
Online Virtual CPUs : 2
Maximum Virtual CPUs : 8
Minimum Virtual CPUs : 1
Online Memory : 1024 MB
Maximum Memory : 16384 MB
Minimum Memory : 512 MB
Variable Capacity Weight : 128
Minimum Capacity : 0.10
Maximum Capacity : 0.80
Capacity Increment : 0.01
Maximum Physical CPUs in system : 16
Active Physical CPUs in system : 8
Active CPUs in Pool : 8
Unallocated Capacity : 0.00
Physical CPU Percentage : 20.00%
Unallocated Weight : 0
```

Based on the given example the LPAR has the following important characteristics when talking about I/O:

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Shared-SMT</td>
</tr>
<tr>
<td>Mode</td>
<td>Uncapped</td>
</tr>
<tr>
<td>Entitled Capacity</td>
<td>0.40</td>
</tr>
<tr>
<td>Minimum Capacity</td>
<td>0.10</td>
</tr>
<tr>
<td>Maximum Capacity</td>
<td>0.80</td>
</tr>
</tbody>
</table>

This uncapped SPLPAR has an **Entitled Capacity** which guarantees currently 0.4 physical Processors, which can be shared if they are not required. The entitled capacity can be changed between a minimum and maximum Capacity value of 0.1 up to 0.8 physical CPUs in the example above.
The partition runs on four logical CPUs because two Online Virtual CPUs with SMT enabled are defined. The uncapped partition can consume up to two physical CPUs since it is capped by the amount of VCPUs. The maximum and minimum Virtual CPUs values allow the amount of online virtual CPUs to be changed from 1 to 8 CPUs online. Here all CPUs of the pool are defined as maximum what guarantees high flexibility. The limitation to currently two online VCPUs protects other SPLPARs with a weight smaller than 128 from being cannibalized and reduces context switches in case of heavily changing CPU assignments.

The Machine has 8 physical CPUs (Active Physical CPUs in System) running (additional CPUs can be in the spare pool or turned off for energy reasons). In this example there are no Dedicated LPARs and only one shared pool. Because the Active CPUs in pool value is as well 8 (This does not give an indication if a Shared Dedicated LPAR exists since those shared Processors are included into the pool).

These unallocated CPUs are in the so called spare pool or turned off for energy reasons. The sum of the number of processor units unallocated from shared LPARs in an LPAR group. This sum does not include the processor units unallocated from a dedicated LPAR, which can also belong to the group. The unallocated processor units can be allocated to any dedicated LPAR (if it is greater than or equal to 1.0 ) or shared LPAR of the group.

The physical CPU percentage is the entitled capacity divided by the number of online CPUs. In this case 8 online CPUs / 0.4% capacity = 20% physical CPU. It is a fractional representation relative to whole physical CPUs that these LPARs virtual CPUs equate to.

The following formula shows the dependencies between entitlement, virtual CPUs and shared pool CPUs:

\[ \text{Entitled Capacity} \leq \text{online virtual CPUs} \leq \text{Active CPUs in Pool} \leq \text{Active Physical CPUs in system} \leq \text{maximum physical CPUs in system} \]
In the next example `lparstat` was used to display the current situation of the LPAR. To monitor CPU shortages it is essential to gather this information before/after and during the shortage occurs.

```
# lparstat 1 4

System configuration: type=Shared mode=Uncapped smt=On lcpu=4 mem=1024 psize=8 ent=0.40

%user %sys %wait %idle physc %entc lbusy vcsw phint
-----------------------------------------------
0.2 1.0 0.0 98.7 0.01 2.2 0.0 484 0
0.0 1.2 0.0 98.7 0.01 2.2 0.0 482 1
0.0 0.7 0.0 99.3 0.01 1.6 0.0 472 0
0.0 0.7 0.0 99.3 0.01 1.5 0.0 458 1
```

As long the `%idle` value is not zero or the consumed entitlement `%entc` is constantly below 100% the LPAR has no shortage since then the shared uncapped LPAR can access additional cycles immediately until all the online VCPUs are used. An `%entc` value of 100% can be fine as long the LPAR can still get additional cycles until all the online VCPUs are used. In that case it is also important to check if this influences the other LPARs negatively.

The amount of `physc` is limited to a value of 2 since only 2 VCPUs are assigned to this LPAR in the given example. That means as long the other LPARs on the shared pool are fine and in the shared pool are unused cycles left the LPAR can access additional cycles as long the `physc` value is below 2.

**mpstat**

Beside `lparstat` AIX® provides the tool `mpstat`. Whereas `lparstat` shows the summary of all logical CPUs of the LPAR, `mpstat` lists each logical CPU separately and can display donated cycles of a dedicated LPAR. That means `mpstat` should be used for Dedicated Shared LPARs.

**topas**

`Topas` provides an offline and an online mode. Online means `topas` runs on command line and prints out the data directly into a file or stdout. Data collected in offline mode are saved to a file as comma separated values and can be pre-processed by `topasout` or copied into an excel spreadsheet.

In the context of this document `topas` is used to get an overview of the whole server, although it has lots of additional functionality. Hence the focus is on `topas -C` (online) and the equivalent `topas -R` (offline). In many cases a snapshot of `topas -C` (the current CEC view) might be the easiest way to start with. It is necessary to collect data during a period of time when the issue shows up and a second time when the system is fine.

```
#topas -C

Topas CEC Monitor Interval: 10 Mon Feb 11 16:26:16
2008
Partitions Memory (GB) Processors
Shr: 15 Mon:58.8 InUse: 46.9 Shr: 5.9 PSz: 8 Shr_PhysB: 0.17
Ded: 0 Avl: - Ded: 0 APF: 7.8 Ded_PhysB: 0.00
```
The offline mode topas -R can collect data up to a 24h period of time and stores it as default in /etc/perf/topas_'date'. With perfagent.tools 5.3.0.40 topasout does not support the -s flag for the CEC view to format the output. Only the CSV version is already available which can be inserted into an excel spreadsheet.

---

**Basic Monitoring of I/O on AIX**

<table>
<thead>
<tr>
<th>Host</th>
<th>OS</th>
<th>M</th>
<th>Mem</th>
<th>InU</th>
<th>Lp</th>
<th>Us</th>
<th>Sy</th>
<th>Wa</th>
<th>Id</th>
<th>PhysB</th>
<th>Ent</th>
<th>%Ent</th>
<th>Vcsw</th>
<th>PhI</th>
</tr>
</thead>
<tbody>
<tr>
<td>is32d2</td>
<td>A53</td>
<td>U</td>
<td>16</td>
<td>15</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>96</td>
<td>0.02</td>
<td>0.40</td>
<td>4.5</td>
<td>588</td>
<td>0</td>
</tr>
<tr>
<td>is3018</td>
<td>A53</td>
<td>U</td>
<td>4.0</td>
<td>3.9</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>97</td>
<td>0.02</td>
<td>0.40</td>
<td>4.3</td>
<td>643</td>
<td>0</td>
</tr>
<tr>
<td>is301v2</td>
<td>A53</td>
<td>U</td>
<td>0.8</td>
<td>0.6</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>97</td>
<td>0.02</td>
<td>0.40</td>
<td>3.8</td>
<td>690</td>
<td>2</td>
</tr>
<tr>
<td>is3017</td>
<td>A53</td>
<td>U</td>
<td>4.0</td>
<td>4.0</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>97</td>
<td>0.01</td>
<td>0.50</td>
<td>3.0</td>
<td>373</td>
<td>1</td>
</tr>
<tr>
<td>is3011</td>
<td>A53</td>
<td>U</td>
<td>4.0</td>
<td>2.1</td>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>97</td>
<td>0.01</td>
<td>0.40</td>
<td>3.5</td>
<td>618</td>
<td>0</td>
</tr>
<tr>
<td>is3048</td>
<td>A61</td>
<td>U</td>
<td>4.0</td>
<td>2.9</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0.01</td>
<td>0.40</td>
<td>2.8</td>
<td>1521</td>
<td>0</td>
</tr>
<tr>
<td>is3031</td>
<td>A53</td>
<td>U</td>
<td>1.0</td>
<td>0.8</td>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>96</td>
<td>0.01</td>
<td>0.20</td>
<td>5.5</td>
<td>532</td>
<td>0</td>
</tr>
<tr>
<td>is3012</td>
<td>A53</td>
<td>U</td>
<td>8.0</td>
<td>2.5</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>98</td>
<td>0.01</td>
<td>0.40</td>
<td>2.6</td>
<td>535</td>
<td>0</td>
</tr>
<tr>
<td>is3046</td>
<td>A61</td>
<td>U</td>
<td>4.0</td>
<td>1.9</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>98</td>
<td>0.01</td>
<td>0.40</td>
<td>2.4</td>
<td>841</td>
<td>0</td>
</tr>
<tr>
<td>is301v1</td>
<td>A53</td>
<td>U</td>
<td>1.0</td>
<td>0.7</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>98</td>
<td>0.01</td>
<td>0.40</td>
<td>2.2</td>
<td>503</td>
<td>2</td>
</tr>
<tr>
<td>is3015</td>
<td>A53</td>
<td>U</td>
<td>1.0</td>
<td>0.8</td>
<td>4</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>98</td>
<td>0.01</td>
<td>0.40</td>
<td>2.1</td>
<td>501</td>
<td>0</td>
</tr>
<tr>
<td>is3019</td>
<td>A53</td>
<td>U</td>
<td>2.0</td>
<td>1.9</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0.01</td>
<td>0.40</td>
<td>2.1</td>
<td>507</td>
<td>0</td>
</tr>
<tr>
<td>is3010</td>
<td>A61</td>
<td>U</td>
<td>1.0</td>
<td>1.0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0.01</td>
<td>0.40</td>
<td>1.8</td>
<td>806</td>
<td>0</td>
</tr>
<tr>
<td>is3016</td>
<td>A53</td>
<td>U</td>
<td>4.0</td>
<td>3.9</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0.01</td>
<td>0.40</td>
<td>1.7</td>
<td>504</td>
<td>0</td>
</tr>
<tr>
<td>is3047</td>
<td>A53</td>
<td>U</td>
<td>4.0</td>
<td>3.7</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>99</td>
<td>0.01</td>
<td>0.40</td>
<td>1.7</td>
<td>406</td>
<td>0</td>
</tr>
</tbody>
</table>

---

The `topas` output has on top a summary of the static information listing the amount of partitions, memory and CPU information. Then followed by a list of the partitions with their most important static as well as dynamic information divided into shared partitions and dedicated partitions. The main difference to `lparstat` is that `topas` lists with one view the entire information of the box and information for each partition. Hence only the differences in the usage are discussed in this section.

**Memory summary section:**

The amount of the displayed memory `Mon` is always smaller than the actual physical amount of memory on the box. Topas shows only the memory that is assigned to partitions. The amount of unassigned memory and memory used by the hypervisor can be seen on the HMC.

**Processors summary section:**

The amount of the physical CPUs is `PSz`. `PSz` is the amount of dedicated CPUs (Ded) plus the amount of shared CPUs (Shr) plus the amount of unassigned CPUs which are not explicitly listed. To calculate how many unused CPUs are on the system the assigned CPUs (Shr and Ded) have to be subtracted from the total amount of CPUs. Reasons for unused CPUs can be to use it for the uncapped mode in the pool, to add more LPARs to this shared pool, and so on.

`%EntC` and `PhysB`:

`PhysB` are the amount of busy physical CPUs whereas the PhysC in `lparstat` shows the number of physical processors consumed what includes idle and I/O wait. Since it is only a snapshot it is very important to get the data during high load to actually see shortages.

**Hint (topas):**

When running topas the windows telnet client is not appropriate. An alternative would be for example the freeware tool PuTTY.
**Other tools**

To know a little bit about the tool sar is very helpful. It displays some information visible in tprof, alstat and emstat and is as well included in PERFPMR. It monitors the major system resources on the local machine.

Enhanced CPU monitoring tools like curt, splat and tprof, have to be used for the following analysis situations by specialists:

- Evaluation of the cache quality: When CPUs are switching often between the LPARS the cache becomes invalid and has to be renewed. One solution to improve the quality of the cache would be for example to use dedicated shared partitions.

- To analyze the CPU usage of the hypervisor if no HMC access is available.

- Analyze CPU consumption when the PowerExecutive is active: When the box starts to save energy the amount of physical CPUs or frequency are changing. This has a direct effect on monitoring not displayed with basic tools yet.

- *tprof* provides detailed information to search for CPU consumers that should not use much CPU like LRU daemon. When using *tprof* the CPU utilization has to be multiplied with the amount of processors.

**Summary of CPU shortage situations**

CPU over commitment:

A physical over commitment is not possible. The HMC does not allow entitling more CPUs than are physically available. CPU over commitment occurs if several LPARs in the uncapped mode exist with the expectation that there will be always cycles from the other LPARs available. That means that on most LPARs the %EntC value and the utilization are often constantly at 100%.

Processor bound workload:

- Processor utilization for the user and system is near 100%.

Processor execution is blocked by an I/O shortage:

- A mentionable amount of time is spent in I/O wait. The I/O wait might be due to slow file access or a lack of memory / paging. In this case not the CPU is the reason.

Unbalanced CPU assignments:

- When CPU shortages are detected and one of the following setups applies it is easy to reconfigure the partition to assign more CPUs:

  - Only very few partitions have CPU shortages
  - CPUs are in the spare pool
  - Uncapped partitions are limited by the amount of online VCPUs. If the maximal VCPUs value is not reached and capacity in the pool is free the tuning can be done online.
  - Some partitions never use their entitlement. These free cycles can be used to increase the entitlement of other partitions.
  - …
**Local storage: File system, Logical and Physical Disks**

Monitoring storage on AIX® is very advanced. On the one hand *iostat* does not deliver information like logical volume information and seek times and on the other hand *filemon* can deliver all necessary information, but unfortunately can not be run for a long time since it is too intrusive. It also requires root authority or AIX 6.1 role based access, if defined. This leads to the situation that disk I/O monitoring in some cases is not possible with the basic AIX® toolset introduced in this paper.

To understand the disk monitoring tools in general, it is important to distinguish between active time (*tm_act*) and utilization:

- The active time indicates the percentage of time the physical disk used for I/O operations excluding time spent by the process. Hence active time is the total time disk requests are outstanding. For example *iostat -D* reports active time and no utilization. The percentage active time of a disk reported by the monitoring commands does not provide any information about the physical utilization of the disk itself. The percentage active time can be 100% while the physical disk is not saturated.

- The utilization is the overall usage of the disk's bandwidth (throughput, number of transactions). The utilization is best measured at the storage server, or in the case of a disk derived from percentage active time in combination with the average number of concurrent I/O's.

**iostat**

The *iostat* command displays only the active time. For basic usage it is recommended to use *iostat -D* to display a list per disk. The provided output is sufficient to apply the following rules:

### Rules of thumb:

**Active time:**
You can say ‘80-90% active time with small number of concurrent I/Os (relative to the queue_depth) are unlikely a problem. This rule again depends heavily on the workload. For example a database server should usually not go well beyond 40% active time for a long period.

**Queuing:**
The avg_serv of reads and writes are high under following circumstances:
- For SCSI 8-10 ms at a queue depth of one. Increasing the depth to 2 the first returns after 10 ms, the second after 20 ms.

The time spend for queuing is high if qfull and/or the queuing_time is above zero.

**Transaction:**
In case the reads and writes are high it indicates an issue with the disk I/O.

Only for systems reading and writing sequentially it is fine, when the tps (transactions per second) value is small and the time spend for read or write time is high.

If the tps and the amount of transferred data figures are high, it is likely that the file is partially in the cache and partially in memory what forces the reads or writes to jump between memory and disk more than once what reduces the performance.
Hint (iostat):

a) To filter all hdisks with tm_act above 70% out of the collected data of the iostat -Dl command call:

```bash
iostat -Dl | awk '/hdisk/ && $2 > 70 { print $0 } '
```

b) To draw a conclusion about the utilization, although iostat does not display a specific value, there is a way to get information by the following two iostat values:

1. Are the read and write avg_serv high?
2. Is the time spent for queuing high?

These two values represent the utilization. The utilization is defined as throughput and transactions. Hence when all of these are high the utilization is high as well.

Hint (PERFPMR):

When using the PERFPMR the same output is in iostat.Dl.

**filemon**

To get more details after an I/O issue has been detected with iostat, filemon can be used. It is not recommended to use filemon directly without knowing that there is an I/O issue due to the fact that it is a very heavy tool, which can not be run for a long time. This makes it hard to monitor shortages that cannot easily be reproduced. Also due to the additional load filemon adds to the machine, the system can crash in very rare situations. To minimize the impact filemon -O ‘Levels’ allows to monitor specified file system levels. Level identifiers for the logical file level (lf), the virtual memory level (vm), the logical volume level (lv), the physical volume level (pv) and all file levels (all) are supported.

Filemon is a trace based tool, what means first a trace has to be collected and then preprocessed. The following is an example how to collect the trace and preprocess it:

1. Start to collect the trace: filemon ‘flags’
2. Stop to collect the trace: trcstop
3. Output is written to fmon.out in the current directory

Hint (PERFPMR):

The PERFPMR dataset contains two ways to get filemon information.

1) generate information out of trace.raw trace:

   Preprocess trace:
   
   ```bash
   trcrpt -r trace.raw > trace.rpt
   ```

   Then run filemon with eg -i and -n flags:
   
   ```bash
   filemon -i trace.rpt -n genames.out -O all
   ```

2) use filemon.sum

   This delivers the same output as filemon with the -i and -n flag.

Remember: Also for PERFPMR filemon output it is valid, that due to the short runtime, it often happens that the filemon output does not collect the data during the time the I/O issue occurs.
Analyze filemon output

The main differentiator from a basic AIX® tool usage between iostat and filemon is the additional information about the logical volumes and the seek time in filemon.

There are three main scenarios for logical volumes to analyze in the filemon output for basic I/O monitoring on local storage:

1) Optimal scenario: The physical discs a logical volume consists of share the load equally and are not fully active (in this case 50% each). That means although the logical volume performs the whole time (100%) I/O operations it always can get the data from the two physical discs without waiting for I/O.

Rule: An active time of a logical volume of 100% does not indicate an I/O issue if the physical discs below constantly provide the requested I/O in time.

2) Physical disc failure scenario: When a physical disk a logical volume depends on becomes hot due to a disk failure, the logical volume can face I/O wait.

Rule: Logical volumes with an active time of 100% should not depend on physical disks with 100% active time since then a simple failure can cause severe I/O issues (disk failures are often reported in the errpt output).
Rule: Also if a logical volume does not hit an high active time I/O issues show up if a physical disk it depends on gets hot due to wrong balancing.

*Figure 8 Assigning physical disks to logical volumes: Wrong load balancing scenario*  

To analyze the logical volumes with *filemon* for the described scenarios the following is a proposal how to get the required data:

1. Filter the most active logical volumes based on the utilization value. The value the utilization can have is between 1.0 (equals 100%) and 0.0 (equals 0%).
2. Match these volumes with the corresponding disks by using for example config.sum in PERFPMR, *lspv -l/M* or *lsv -l/m* to be able to check all physical volumes whether they are hot or not.
3. Look into the detailed physical and logical volume stats for:
   - ongoing high utilization
   - long read and write average times
   - long seek times

**Rule of thumb:**

*Whereas seek, read, and write times depend strongly on how and which storage is attached are relative. But an ongoing utilization of 0.9 – 1.0 is a clear sign of an I/O issue in every case.*

In the following the recommended information is applied to the introduced scenarios:

**Optimal scenario:**

- High utilized logical volumes (1.) do not depend on high utilized physical disks (2.).

**Physical disk failure scenario:**

- High utilized logical volumes (1.) depending on high utilized physical disks (2.) which experience (3.).

**Wrong load balancing scenario:**

- Low and normal utilized logical volumes (1.) depend on the same physical disk (2.). If the shared physical disk is highly utilized and the logical volume is facing (3.) it is scenario 3.

**Hint (filemon):**

*Up to AIX 5.3 it is important to know that the distinction between utilization and tm_act is not as is today.*

*For I/O monitoring all statistics of filemon regarding files and VM segments can be ignored, but can be helpful for none I/O related performance analysis which are not in the scope of this paper.*

netpmmon

How to detect hot files with netpmmon will be described in the NFS section later.

**Network**

The network consists of the following layers which will be discussed in the corresponding subchapters in detail:

- Protocols
- Interfaces
- Adapter
- Packet transfer

To test the network for I/O issues it is best to check protocol, interface, adapter and packet transfer in this given order. The reason for that ordering is the dependency the layers have between each other. The following examples are describing the dependencies:

**Dependency 1)**

When traversing the stack top down all layers not showing an issue are fine until the first layer showing an issue is discovered. For example if the interface shows problems the root cause can never be the protocol layer and has therefore to be searched in the interface layer or below (marked orange).

![Figure 9 Network stack dependency 1](image-url)
Dependency 2)

When detecting the first layer with an I/O issue the rest of the stack has to be traversed till the first layer not showing an I/O issue is found. That means that the root cause is likely to be the layer above the current one. For example if the interface layer was the first layer showing the I/O issue and the packet transfer layer is completely fine, whereas the two layers in between have issues, the adapter layer is likely a candidate for the root cause.

![Network stack dependency 2 diagram](image)

*Figure 10 Network stack dependency 2*

The tools used for the network I/O analysis are `netstat` and `entstat`. Both are collecting the statistics from system start. Hence the delta of two snapshots erases old information what builds the base of the analysis.

**Protocols**

The `netstat -s` command shows a list of all protocols with their statistics. In case the system does not use all protocol types `netstat -s -s` shows only used protocols to reduce the output.

To analyze the `netstat` output the following recommendations can be given:

- Check if the load in packets per second is fine. This max load depends on the used cable and adapter characteristics.
- Attributes containing the word “bad” should be zero. None zero values can indicate an issue.
  - Send statistics
  - No retransmission should occur. When retransmission shows up the value should be marginal in comparison to the packets that have been sent in total. To analyze retransmission issues in depth the `iptrace` tool (advanced) has to be used. `netstat` can only be used to get a green/yellow/red performance indicator.
- Receive statistics:
  - If there are no incoming packets (`received = 0`) the adapter and mbufsize might produce the issue.
  - Out of order packets are a sign for packet loss on the way or a sender issue. A sender issue could for example be a socket buffer overflow which results in packet loss. Based
on experience it is more likely that packets get lost than the sender of the packets is the root cause.
  o A high delta value of window probe packets between two snapshots indicates a bad configuration of the window size on the client or the server side.
  o A negative delta value of the window probe packets indicates that the sever sends more and more probe packets what is a server side issue with the protocol configuration.

Hint (PERFPMR):
The data set delivered from PERFPMR already provides two snapshots. This makes a second run unnecessary.

**Network Interfaces**
The interface information is delivered by using the flags `-in`, `-D` and `-v` of `netstat`, which are showing statistics per interface.
The `-in` flag only shows the state of the configured interfaces. Hence it is perfect for an initial check.
Using `-D` displays incoming and outgoing packets of each layer in the communications subsystem along with packets dropped per layer. This information helps to narrow down the issues by looking into the device statistics, driver, demux (protocol) and the amount of dropped packets.
Finally the `-v` flag gives detailed information for issues seen in the `-D` output.
In general it is recommended to check `netstat` outputs for the following:
  • All attributes containing the word “bad” should be zero.
  • Incoming and outgoing packet errors should be zero.
    o Are dropped packets due to an adapter issue?
    o Otherwise check the interface’s configuration and the driver.

Hint (netstat):
Basic tuning recommendations can be found on the AIX Information Centre homepage:  
tungd/netstat_in.htm

**Network Adapter**
The `entstat` command requires the name of a specific Ethernet device driver. Hence the command might be run more than once.
The recommended points to look for in the `entstat -d `adapter-name` output are:
  • Number of dropped packets should be marginal. If a lot of packets are dropped `entstat` delivers following possible reasons for the dropped packets:
    o Wrong CRC checksum.
    o No resource errors. For example incoming packets can not be stored in the queue.
    o Max Collision/Late Collision Errors with Half duplex.
• A high number of bad packets leads to the conclusion that the physical network has a problem (broken or unplugged cable, …).

• The summary section of the adapter statistics gives information about the health of the adapters.
  
  o The settings on the switch and the adapter have to correlate to enable the adapter to send the packets through the switch. The information about the switch is not visible from within AIX®. As a rule the tuning on both has to be the same for:
    
    - The selected media speed.
    - "Auto negotiation" on for Ethernet connections.
    - Jumbo frames on or off.

• Software transmit queue overflows resulting in dropped packets are a sing for a too small send-queue.

• The protocol totals show data per protocol which can be helpful to narrow a problem.

**Hint: (PERFPMR):**

In PERFPMR the corresponding values are in the file netstat.int and not as expected in entstat.

### Packet transfer statistics

To get a general health check of the adapters, cables and how the connection performs in the sense of bytes per second, a memory to memory copy between the client and the server can be performed using `ftp`. This excludes side effects from disk access, CPU and so on. If the transfer rate is good or bad depends on the used adapters and cables.

The following test has to be performed from the client as well as from the server side:

```
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
```

An example output of this command looks like following:

```
ftp> put "|dd if=/dev/zero bs=32k count=10000" /dev/null
200 PORT command successful.
150 Opening data connection for /dev/null.
10000+0 records in.
10000+0 records out.
226 Transfer complete.
327680000 bytes sent in 2.825 seconds (1.133e+05 Kbytes/s)
local: |dd if=/dev/zero bs=32k count=10000 remote: /dev/null
```

For all further analysis the advanced tools `iptrace`, `ireport` and `ipfilter` have to be used. For completeness short annotations to those tools:

Using `iptrace` (advanced) is critical when trying to determine packet loss without checking the layers above. Also a good knowledge about the protocol stack must be available to interpret the data. It is good to know, that `iptrace` traces can not track all packets during very high load and the untracked packets are then added to the „lost packets“ section.

Also the output of `ireport` as a filter tool for `iptrace` data with the flags `-srn` can be used. The data shows statistics about the connections per packet such as source IP and port, destination IP and port, packet information that includes also number of hops, response time, etc. .
Beside *iptrace* and *ipreport*, *ipfilter* is a third tool which generates table views out of the output of *ipreport*.

More about network performance analyze tools can be found here:

tungd/network_perf_analysis.htm

**Hint (PERFPMR):**

In PERFPMR the iptrace report can be generated by using *iptrace.sh* -r in the directory of where the PERFPMR traces are.

The reports can be generated as following:

```
ipreport: ipreport -srn iptrace.raw > iptrace.ipreportSRN
ipfilter: ipfilter [flags] iptrace.ipreportSRN
```

### NFS Client

An NFS environment consists of a client and a server side. The NFS server only has I/O problems when the client has them as well. Therefore the client should always be analyzed first. On the NFS client it is suggested to check the local resources first as described in the earlier sections, followed by the NFS specifics if necessary. The following figure shows the recommended order of analyzing NFS related I/O issues.

**Recommended top-down NFS trouble checklist:**

- Check the biod settings of the client.
- Verify that the network connections are good as described earlier.
- Although there are other NFS daemons verify that the inetd, portmap, and biod daemons are running on the client for example with the *ps* command: `ps -ef | grep 'name'`.
- Verify that a valid mount point exists for the file system being mounted. For example the *mount* tool can be used for that purpose.
- Verify that the server is up and running by running the following command at the shell prompt of the client: `# /usr/bin/rpcinfo -p 'server name'`
- Verify if the NFS mount has hot files or disks.

Further NFS performance monitoring hints can be found at:

tungd/nfs_perf.htm
**Validate biod settings**

The biod handles the client's sequential read and write requests to the server and by that the settings only apply to the client. biod based bottlenecks can occur when many processes are writing on a slow servers or if the client has not enough biod to handle the read and write requests.


These settings are valid for all standard cases and recommended as an initial value for tuning purposes.
Trace based analysis of biod issues

The trace based analysis is useful to tune the biod settings. The trace can be found in /etc/trcfmt. Example 1 shows a trace with an insufficient amount of biod whereas Example 2 also shows a trace with problems, although enough biods have been applied.

Example 1: insufficient number of biod daemons

Example 2: NFS server(s) can not deal with amount of I/O

Basic Monitoring of I/O on AIX
Basic Monitoring of I/O on AIX

During the read long biod runtimes (white boxes) occur and are the reasons for the I/O performance.

The time from making the kbiod runnable (setrq: cmd=kbiod) until it becomes dispatched (dispatch: cmd=kbiod) is the amount of time the NFS client needs to dispatch the kbiod thread. This time usually is very short (few microseconds). A long time period between setrq and dispatch (several milliseconds and more) can indicate a load issue on the client which then has to be investigated in.

Example 2 shows a piece of a trace with high kread (grey boxes: “kread” to “return from kread”). During the read long biod runtimes (white boxes) occur and are the reasons for the I/O performance.
The time between dispatching the kbiod until it makes the cp command runnable includes the time the NFS server takes to respond to a client request. A long period here indicates an issue on the NFS server side, which should be investigated before analysing the client further. The following are the related lines from the example above:

```
106     kbiod  2  450781  0.023792  dispatch: cmd=kbiod pid=184410
         tid=450781 priority=60 old_tid=69667
         old_priority=255 CPUID=2 [84 usec]
11F     kbiod  2  712805  1  0.094713  setrq: cmd=cp pid=307386 tid=536635
```

**Detect hot files and disks with netpmon**

The *netpmon* tool provides two possible approaches to find the hot files or processes:

a) If a name or id to look for is available, like an SAP job name, a PID or a file name on AIX level, it is also in the output of *netpmon*.

b) If no name or ID is available, *netpmon* provides general statistics for NFS by process and by file where slow files and processes can be determined.

Example for approach b):

The “Detailed NFS client Statistic (by Process)” section in the *netpmon* output gives information about I/O wait for each process. Hot processes can be identified by long average read times in the *read times (msec): avg ...........* column.

By analyzing the other detailed NFS statistics as well for the average times it is easily possible to narrow the problem down when knowing the hardware. The outcome can be:

- A slow file
- A slow process
- A slow server


**Rules of thumb (netpmon):**
Every average of read times below 5 ms should be fine.
Due to the fact that netpmon only measures the queuing time from the client side when using external storage this value should be below 5ms. Whereas when writing on the local storage a higher (8-10ms) read time than 5 seconds can occure.

**Hint (PERFPMR):**
Before applying netpmon the trace provided in the PERFPMR output has to be preprocessed with trcrpt:
```
trcrpt -C all -r trace.raw > trace.tr
```
Then netpmon can be run on base of the output of trcrpt:
```
netpmon -i trace.tr -n gennames.out -O all > trace.netpmon
```
The output will be written in a file called trace.netpmon. In case no specific client or server side traffic occurred only the detailed view and no summaries by client or server are generated.
In addition PERFPMR provides filemon.sum which is showing read-times per process id but only during a very short period of time.

**Virtual I/O Client**

This section introduces how a basic setup looks like. A virtual I/O client is a client where the adapters to the network and/or storage are virtualized by a virtual I/O server (VIOS) which always resides on the same physical box. Therefore the already described methods for I/O monitoring can be applied as is.

The I/O characteristics between a VIO client and a VIOS are different compared to NFS. Both, the client and the server, are always on the same physical box, since this is the only way to share I/O resources between clients. For load balancing and reliability reasons usually two VIO server reside on the same box.

The top-down approach starts again with the client’s local resources followed by the adapter to the VIO server and finally the VIOS from bottom-up. First the device drivers have to be checked, followed by the queue depth of the hard-disk configuration. When the problem can not be determined on the VIOS, it becomes a client to the iSCSI attached storage (not common) or the fabric it is attached to. Both ways of attaching storage are beyond the scope of this whitepaper and by that not further described.
Analysis of AIX® Server

Now after all steps of the client analysis have been applied and no issue on the client side has been found the problem seems to be server related. Hence the server analysis has to be started. Overall the server analysis is more or less the same as the client analysis. Therefore for methods equal to the client analysis are not covered in this section a second time.

The AIX® server analysis starts with the connection between server and client followed by the proposed bottom-up approach (Figure 13). If the server does not cause the problem it gets the client credentials to another server it is attached to and the analysis is to be continued.
Network

The server sided network has only to be checked if on client side network issues occur. On the server the analysis is applied bottom-up:

1. Check of the physical cable with `ftp` memory to memory copy.
2. Check of the adapters with `entstat`.
3. Check of the interfaces with `netstat`.
4. Check of the protocol layer with `netstat`.

Memory and CPU

Since memory and CPU are local the same analysis as on the client side applies.

Logical and Physical Discs

For disk I/O again only the local disks can be checked with basic tools using `iostat` and `filemon` as described in the client section. This excludes monitoring of SAN Volume Controller, other fiber channel attached storage or iSCSI attached devices.
**NFS Server**

The tool `netpm` also delivers the counterpart to the client analysis for CPU, I/O of network devices, NFS and sockets. In addition we look at the associated information of the client, server and processes and the associated response time. Depending on the application, some can withstand very long response times and others require very short response times.

In addition to the analysis done with `netpm` the following steps are proposed:

- Check the NFS tunable with `nfs -L` which provides a list showing the defaults and the current setting.
- Compare the NFS server options with the client mount option (`mount` shows the virtual file system type in the column `vfs`)

**Virtual I/O Server**

VIO server should perform without changing any of its default tuning settings. The only changes typically done are adjusting the `queue_depth` parameter for the `hdisk` - typically Fiber Channel (FC) connected storage servers; some also require installing third party device drivers which might adjust the `hdisk` settings automatically. The IBM® tuning recommendation of a VIOS is more a must than a proposal. A VIOS should always be tuned with the default values.
Summary

Performance depends on customers’ expectations. Therefore this document can not be seen as a black and white handbook it is more a collection of initial tips and the basic tools shipped with AIX 5.3.

We discussed that performance issues can come up due to any changes in:

- Hardware configuration - Adding, removing, or changing configurations such as how the disks are connected
- Operating system - Installing or updating a file set, installing PTFs, and changing parameters
- Applications - Installing new versions and fixes, configuring or changing data placement
- Tuning options in the operating system, RDBMS or application
- Any other changes or accidents like broken cables and so on

Furthermore, the client-server approach has been introduced with the three main points:

- If the client is performance wise fine the server is fine as well.
- If you have no clue where to start begin with the client from top-down and if you do not find anything go to the server and begin bottom-up.
- If the server is fine the server is client to a server. In this case restart your investigations.

Monitoring I/O on AIX® systems is an art one must understand. The introduced tools, which are a selected subset of the tools AIX 5.3 provides, are sufficient for the basic I/O monitoring. For further details trace based tools have to be used. To analyze performance it is important to maintain a performance history to identify changes. Always collect data before, after and during an I/O issue and narrow down the issue with the basic tools.
Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM System i Information Center
  http://publib.boulder.ibm.com/iseries/

- IBM System p Information Center
  http://publib.boulder.ibm.com/infocenter/pseries/index.jsp

- IBM Publications Center

- IBM Redbooks
  www.redbooks.ibm.com/
  - AIX 5L Practical Performance Tools and Tuning Guide
  - AIX 5L Performance Tools Handbook
  - Problem Solving and Troubleshooting in AIX 5L

- Internal – Search Database for CVSM and PMRs
  https://techlink.austin.ibm.com/psdb/systemp

  Web address

- “CCMS Enhancements” presentation from Olaf Rutz Interlock 2007


About the authors

Katharina Probst - probst@de.ibm.com
Dirk Michel - dirkm@us.ibm.com

Acknowledgements

Olaf Rutz (IBM, Germany)
Walter Orb (IBM, Germany)
Georg Leffers (SAP)
Augie Mena (IBM, Austin)
Trademarks and special notices

© Copyright IBM Corporation 2008. All rights Reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM and the IBM logo are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

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

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

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

ABAP, SAP EarlyWatch, SAP GoingLive, SAP NetWeaver, SAP, and SAP logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries.

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

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages.

IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.